REAL TIME CONTROLLER

โดย
นายสมคิด หรรษานิมิตกุล
นายสังคม รูปแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาเทคนิคอุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2534

หัวข้อปริญญานิพนธ์ REAL TIME CONTROLLER

โดย นาย สมคิด หรรษานิมิตกุล
นาย สังคม รูปแก้ว
ภาควิชา เทคนิคอุตสาหกรรม
อาจารย์ที่ปรึกษา อาจารย์ อรรถสิทธ์ หล่าสกุล

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ ฉบับนี้เป็นส่วนหนึ่งของ การศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์

คณะกรรมการสอบปริญญานิพนธ์

.....อาจารย์ที่ปรึกษา
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

REAL TIME CONTROLLER

นาย สมคิด หรรษานิมิตกุล
นาย สังคม รูปแก้ว

อาจารย์ อรรถสิทธิ์ หล่าสกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2534

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นเครื่องควบคุมอุปกรณ์ไฟฟ้า โดยการนำเอาไมโครโปรเซสเซอร์ กับไอซี Real time clock มาทำการควบคุมอุปกรณ์ไฟฟ้าต่างๆ เครื่องควบคุมอุปกรณ์ไฟฟ้าเครื่องนี้สามารถควบคุมได้ 8 Channel การโปรแกรมของแต่ละ Channel จะมี Function ของการทำงานให้เลือกถึง 4 Function โดยในแต่ละ Channel จะมีการทำงานของโปรแกรมเป็นอิสระต่อกัน การทำงานของโปรแกรมควบคุมยังสามารถกำหนดกระแสของโหลดได้ด้วย Soft wear ซึ่งจะเป็นประโยชน์อย่างมากในการป้องกันอันตรายที่อาจจะเกิดขึ้นได้ และถ้ามีข้อผิดพลาดเกิดขึ้นโปรแกรมควบคุมก็สามารถบอกข้อผิดพลาดที่เกิดขึ้นได้ ความละเอียดของโปรแกรมควบคุมมีความละเอียดถึงระดับวินาที การแสดงการทำงานของเครื่องจะแสดงโหมดการทำงาน, เวลาของเครื่อง, สภาวะของโหลด และข้อผิดพลาดต่างๆ ที่เกิดขึ้นทางจอ LCD ขนาด 40 ตัวอักษร ทำให้ง่ายต่อการใช้งาน ซึ่งทั้งหมดนี้ถูกรวบรวมในปริญญานิพนธ์ฉบับนี้

REAL TIME CONTROLLER

SOMKID HUNSANIMITKUL

SANGKOM ROOPKAEW

MR. ATTASIT LASAKUL

(ADVISOR) '1991

ABSTRACT

This thesis is device control electric equipment by Mircoprocessor with real time clock to control electric device can control output 8 channel. Program of channel will have function operate give selected 4 function. Reach channel will have operation of program free-dom . Program can control limit current circuit by software . It is usely a lot of safety system electric.

Program can set time in 1 second minimum. Working of circuit is control by software which control all circuit and have screen LCD type. Detail of circuit and software arein this thesis.

สารบัญ

บทที่ 1

บทนำ

จุดประสงค์

1. พัฒนาเครื่องที่มีอยู่ในท้องตลาดให้มีขีดความสามารถในการทำงานสูงขึ้น
2. ใช้ควบคุมเครื่องใช้ไฟฟ้าต่างๆ เพื่อเป็นการประหยัดพลังงาน
3. สามารถนำไปประยุกต์ใช้ในงานด้านอื่นๆได้

ขอบเขตการทำงาน

1. ต่อ load ได้ 8 channal ✓
2. โปรแกรมในแต่ละ channel ๆละ 100 โปรแกรมและเป็นอิสระต่อกัน
3. ในการใช้นาฬิกาจาก REAL TIME CLOCK แม้ว่าไฟ main ดับนาฬิกายังเดินต่อได้
4. ควบคุมกระแสของ load ได้ให้ load ทำงานอยู่ในช่วงกระแสที่ทำการตั้งไว้
5. ตรวจสอบได้ว่า load ที่เครื่องสั่ง ON แล้วทำงานหรือไม่ได้
6. สามารถทำการ ON/OFF load ได้โดยตรง
7. สามารถทำการโปรแกรม เครื่องได้จากเครื่อง IBM PC ทำให้ง่ายต่อการโปรแกรมเครื่องหลาย ๆ โปรแกรม
8. มี FUNCTION ของโปรแกรมมีทั้งหมด 4 FUNCTION ดังนี้

   FUNCTION 1 ทำงานเหมือนกันทุกวัน

   FUNCTION 2 ทำงานเหมือนกันทุกสัปดาห์

   FUNCTION 3 ทำงานเหมือนกันทุกเดือน

   FUNCTION 4 ทำงานเหมือนกันทุกปี

FEATURES

1. ใช้อุปกรณ์ไฟฟ้าที่ใช้ไฟ 220 V_{ac}
2. จอแสดงผลเป็นจอ LCD สามารถแสดงผลได้ 2 บรรทัดละ 20 ตัวอักษรทำให้ดูง่ายโดยจะแสดง วัน-เดือน-ปี และเวลากับสภาวะของ load ว่า load อยู่ในสภาวะใด
3. คีย์บอร์ดขนาด 24 คีย์ใช้งานง่ายและมีหลาย FUNCTION
4. สารถทำงานแก้ไขโปรแกรมที่มีอยู่ได้ และทำการลบออกได้
5. มีระบบ back up data ของโปรแกรมในขณะที่เกิดไฟดับทำให้โปรแกรมอยู่
6. สามารถควบคุมอุปกรณ์ได้โดยตรงจากคีย์บอร์ด

การประยุกต์ใช้งาน

1. ใช้ควบคุมกระบวนการผลิต โดยการตั้งเวลาการทำงานของเครื่องจักรให้มีการคำนวณอย่างต่อเนื่อง หรือตามลำดับทำให้ง่ายต่อการควบคุม
2. ใช้ตั้งเวลาล่วงสำหรับอุปกรณ์บางอย่าง เช่น เครื่องจักรที่ต้องการอุ่นเครื่องก่อนการทำงาน
3. สำหรับการประหยัดพลังงาน โดยการปิดอุปกรณ์ในเวลาที่ไม่ได้ใช้งาน เช่น เครื่องทำความเย็น เครื่องจักรที่ยังไม่ได้ใช้งาน
4. ควบคุมอุปกรณ์ที่ต้องเปิดปิดตามเวลา เช่น การเปิดปั๊มน้ำรดต้นไม้การเปลี่ยนน้ำในบ่อปลา เครื่องให้ออกซิเจนในบ่อปลาและบ่อกุ้ง
5. ใช้สำหรับการเปิดอุปกรณ์ในเวลากลางคืน และปิดเองในตอนเช้า เช่น การเปิดไฟป้ายโฆษณา การเปิดไฟหน้าประตูบ้าน

โครงสร้างของ FUNCTION PROGRAM

FUNCTION 1 เป็นฟังก์ชั่นที่ให้โปรแกรมทำงานทุกวันเหมือนกัน โดยโปรแกรมจะทำการตรวจสอบเฉพาะ ชั่วโมง นาทีและวินาทีเท่านั้น

FUNCTION 2 เป็นฟังก์ชั่นที่ทำงานในรอบ 1 อาทิตย์ โดยสามารถที่จะกำหนดวันหยุดในสัปดาห์ได้ โปรแกรมจะทำการตรวจสอบเหมือนใน FUNCTION 1 แต่จะเพิ่มการตรวจวันในรอบสัปดาห์ด้วย

FUNCTION 3 เป็นฟังก์ชั่นที่ทำงานในรอบ 1 เดือน โดยสามารถที่จะกำหนดวันหยุดในรอบเดือนได้ โปรแกรมจะทำการตรวจสอบเหมือนใน FUNCTION 1 แต่จะเพิ่มการตรวจวันที่ในรอบเดือนด้วย

FUNCTION 4 เป็นฟังก์ชั่นที่ทำงานในรอบ 1 ปี โดยสามารถที่จะกำหนดวันหยุดในรอบปีได้ โปรแกรมจะทำการตรวจสอบเหมือนใน FUNCTION 1 แต่จะเพิ่มการตรวจวันที่และเดือนในรอบปีด้วย

โครงสร้างของ MODE ใน MANUAL

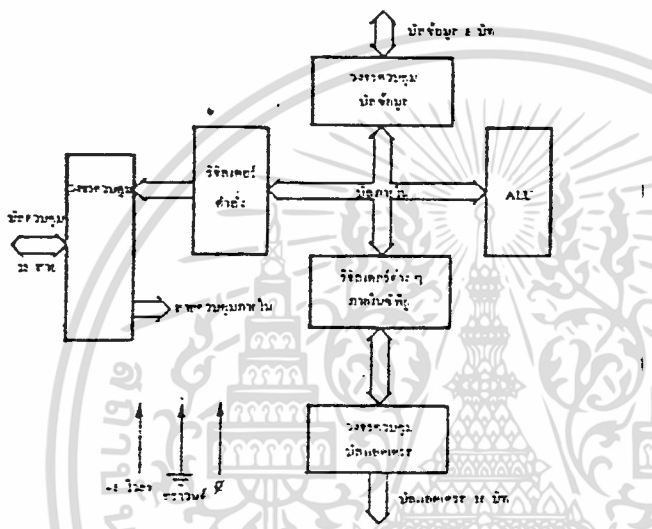
MODE 1 ทำการเปิด/ปิด load โดยตรง เมื่อถึงเวลาที่โปรแกรมตั้งเอาไว้ load ก็จะทำงานตามโปรแกรมนั้นทันที

MODE 2 ทำการเปิด/ปิด load โดยตรง โดยไม่คำนึงถึงโปรแกรม

บทที่ 2

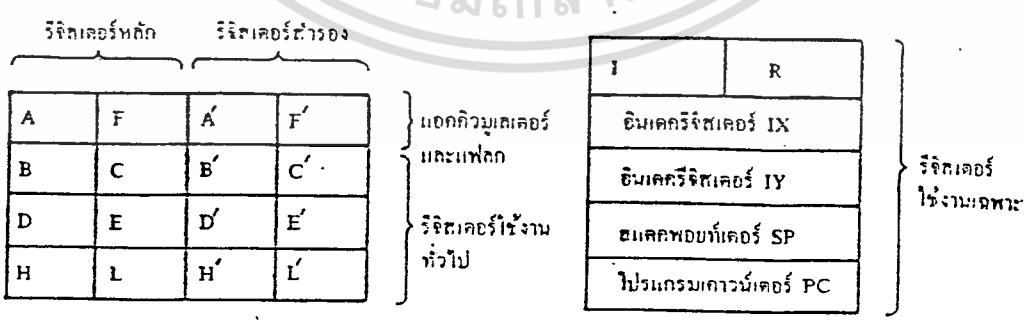
ส่วนควบคุมการทำงานของระบบ

โครงสร้างของซีพียู z-80 พัฒนามาจาก 8080 แต่มีรายละเอียดแตกต่างเพิ่มเติมอีกหลายประการด้วยกัน บล็อกไดอะแกรมรูปที่ 2.1 แสดงให้เห็นโครงสร้างของซีพียู z-80 ที่บรรจุลงในแอลเอสไอขนาด 40 ขา



รูปที่ 2.1
บล็อกไดอะแกรมซีพียู Z-80

โครงสร้างภายในของ z-80 ซีพียูประกอบด้วยรีจิสเตอร์ภายในที่สามารถเขียนและอ่านได้ถึง 208 บิท โดยแยกเป็นกลุ่มของรีจิสเตอร์ขนาด 8 บิท 18 รีจิสเตอร์ และรีจิสเตอร์ขนาด 16 บิทอีก 4 รีจิสเตอร์ โดยมีชุดรีจิสเตอร์แสดงได้ดังรูปที่ 2.2



รูปที่ 2.2 แสดงรีจิสเตอร์ต่างๆที่มีอยู่ใน z-80

รีจิสเตอร์หลักที่ใช้งานทั่วไป

รีจิสเตอร์ในกลุ่มแรกคือ A,F,B,C,D,E,H,L เป็นรีจิสเตอร์ขนาด 8 บิท ที่ใช้งานทั่วไป โดยสามารถประกอบรวมเป็นคู่รีจิสเตอร์ได้ คือ AF,BC,DE และ HL โดยใช้งานในลักษณะของรีจิสเตอร์ขนาด 16 บิท การกระทำภายในซีพียูอาจจะอาศัยเพียงรีจิสเตอร์เดียวหรือกระทำเป็นคู่รีจิสเตอร์ได้โดยที่ A คือแอคคิวมูเลเตอร์ F คือ แฟลก แฟลกของ Z-80จะมี 6 ตัวจริงใช้เพียง 6 บิท แต่ Z-80 อาศัยการเพิ่มบิทอีก 2 บิทและกลายเป็นรีจิสเตอร์ F ซึ่งสามารถได้รับการเซทรีเซทการกระทำตามคำสั่งทางคณิตศาสตร์ หรือลอจิกได้ และสามารถใช้ F เหมือนรีจิสหนึ่งซึ่งเมื่อรวมกันกับ A แล้ว จะกลายเป็นรีจิสเตอร์ขนาด 16 บิทได้

กลุ่มรีจิสเตอร์สำรอง

เป็นกลุ่มรีจิสเตอร์ที่สามารถเก็บข้อมูลได้ โดยเป็นตัวเก็บข้อมูลที่มาจากรีจิสเตอร์หลัก รีจิสเตอร์ชุดนี้จึงมีด้วยกัน 8 ตัว คือ A',F',B',C',D',E',H',L' รีจิสเตอร์เหล่านี้ใช้ในการเก็บข้อมูลชั่วคราว ในการที่ต้องการใช้รีจิสเตอร์หลักทำงานอย่างอื่นก่อน ดังนั้นรีจิสเตอร์เหล่านี้จึงไม่สามารถกระทำทางคณิตศาสตร์และลอจิกได้

กลุ่มรีจิสเตอร์ที่ใช้งานเฉพาะอย่าง

โปรแกรมเคาวเตอร์ (PC-Program counter) *เป็นรีจิสเตอร์ขนาด 16 บิทที่เป็นตัวกำหนดตำแหน่งของโปรแกรมในขณะสภาวะการกระทำการเฟทช์* โดยขณะทำการเฟทช์ค่าอยู่ในโปรแกรมเคาวเตอร์จะไปปรากฏอยู่ที่แอดเดสบัสเพื่อชี้ไปยังตำแหน่งในหน่วยความจำให้ซีพียูอ่านคำสั่งมาตีความหมาย ค่าที่อยู่ในโปรแกรมเคาวเตอร์จะเพิ่มค่าขึ้นได้อย่างอัตโนมัติหลังการกระทำการเฟทช์ แต่ถ้าหากซีพียูกระทำคำสั่งให้ข้ามไปยังตำแหน่งอื่น ค่าแอดเดรสที่จะกระโดดข้ามนั้นจะโหลดเข้ามาในโปรแกรมเคาวเตอร์ได้อย่างอัตโนมัติ

สแตคพอยเตอร์ (SP-Stack pointer) เป็นรีจิสเตอร์ที่มีขนาด 16 บิท ที่ใช้สำหรับชี้ไปยังแอดเดรสชั้นบนสุดของสแตคที่อยู่ในRAM โดยส่วนของสแตคมีลักษณะโครงสร้างเป็นหน่วยความจำแบบ*เก็บทีหลังเรียกออกได้ก่อน* ข้อมูลในสแตคอาจได้รับ*การพุช*หรือ*พอพ*มาจากข้อมูลรีจิสเตอร์ภายในซีพียู ในที่นี้ยังเป็นส่วนช่วยใน*การกระทำอินเตอร์รัพท์ และการเรียกโปรแกรมย่อย* กล่าวคือในการอินเตอร์รัพท์ค่าของโปรแกรมเคาวเตอร์จะได้รับการเก็บรักษาไว้ในสแตค ครั้นเมื่อโปรแกรมกลับจากอินเตอร์รัพท์ไปกระทำยังโปรแกรมหลักก็จะนำค่าสแตคกลับเข้ามายังโปรแกรมเคาว์เตอร์ใหม่ ในทำนองเดียวกัน การกระโดดไปทำยังโปรแกรมย่อยก็เช่นเดียวกัน ดังนั้นการกระทำ

ในรูปของอินเตอร์รัพท์หรือโปรแกรมย่อย สามารซ้อนกันได้ไม่มีที่สิ้นสุด

อินเดครีเจสเตอร์ (IX,IY-index register) ซีพียู Z-80 มีอินเดครีจิสเตอร์ขนาด 16 บิท 2 ตัว แต่ละตัวทำหน้าที่เป็นตัวเก็บแอดเดรสฐาน (base address) เพื่อทำหน้าที่*อ้างแอดเดรสแบบอินเดคแอดเดรสซึ่ง*(index addressing) ในโหมดนี้จะรวมกับข้อมูลที่ติดมากับคำสั่งอีก 8 บิท เพื่อเป็นตัวกำหนดแอดเดรสให้กับคำสั่งข้อมูลที่ติดมากับคำสั่งนี้ เราเรียกว่า *ดิสเพลซเมนท์* (displacement) ซึ่งจะเก็บในรูปของตัวเลข2's คอมพลีเมนท์

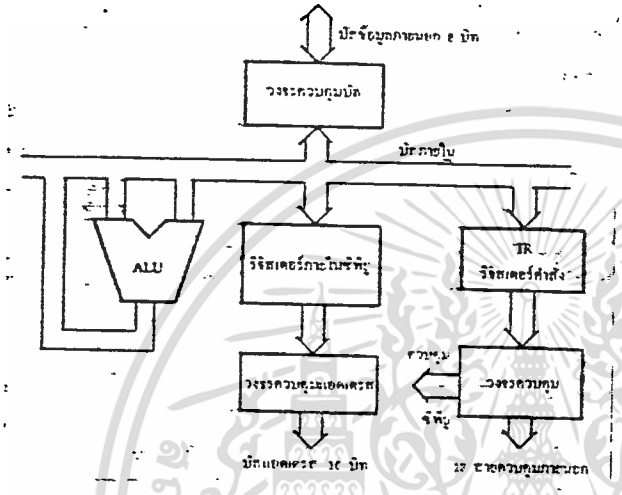
อินเตอร์รัพท์เพจแอดเดรสรีจิสเตอร์ (I-Interrupt page address register) การอินเตอร์รัพท์ของZ-80มีหลายโหมด และโหมดหนึ่งที่ทำให้การอินเตอร์รัพท์มีประสิทธิภาพสูง กล่าวคือมันสามารถอ้างแอดเดรสโดยทางอ้อมไปกระทำโปรแกรมในที่ใดก็ได้ในหน่วยความจำ โดยอาศัยค่าในรีจิสเตอร์Iรวมกับค่าที่ส่งมากับอุปกรณ์เพอริเฟอรัลอีก 8 บิทชี้ไปยังหน่วยความจำ เพื่อนำค่านั้นมาโหลดเข้าในโปรแกรมเคาเตอร์เพื่อกระทำต่อไปด้วยวิธีการนี้จึงสามารถกระโดดเข้าไปทำที่ส่วนใดก็ได้ในหน่วยความจำ

รีจิสเตอร์รีเฟสหน่วยความจำ (R-memory reflesh register) การต่อซีพียูกับหน่วยความจำนั้นโดยปกติจะต่อกับหน่วยความจำชนิดสเตติคได้โดยง่าย อย่างไรก็ดี *ชนิดไดนามิคที่ต้องการรีเฟรชมีราคาถูกกว่า มีความหนาแน่นสูงกว่า* Z-80 ให้ข้อดีกว่าประการหนึ่งคือมันสามารถให้การรีเฟชหน่วยความจำได้อย่างอัตโนมัติ โดยค่าใน R รีจิสเตอร์จะเพิ่มขึ้นอีก 1 ทุกครั้งที่มีการกระทำการเฟชคำสั่ง และจะส่งออกไปยังแอดเดรสบัสในส่วนบิทที่มีนัยสำคัญต่ำกว่า จังหวะของการส่งนี้จะเป็นจังหวะเดียวกับที่ซีพียูส่งสัญญาณรีเฟชออกมา เราสามารถกำหนดค่าให้กับรีจิสเตอร์ R ได้ แต่ค่าในรีจิสเตอร์นี้จะเรียกมาใช้ทางคำสั่งโดยตรงไม่ได้

แอคคิวมูเลเตอร์ (accumulater) **และแฟลก** (flag) เป็นรีจิสเตอร์ที่ใช้เป็นหลักในการเป็นตัวโอเปอร์แรนด์สำหรับกระทำทางคณิตศาสตร์และลอจิก จะมีเพียง 8 บิทเรียกว่า *แอคคิวมูเลเตอร์ (accumulator)* การกระทำในส่วนของหน่วยคณิตศาสตร์และลอจิกย่อมเกิดเงื่อนไขได้หลายอย่างที่จะต้องแสดงสถานะภาพของเงื่อนไขเหล่านั้น เช่นเงื่อนไขผลลัพท์เป็นศูนย์ ผลลัพท์เป็นบวกหรือลบมีตัวทดหรือมีตัวยืมในการกระทำทางคณิตศาสตร์ แสดงเงื่อนไขพาริตี้คู่หรือคี่ ฯลฯ สิ่งเหล่านี้จะให้ผลลัพท์แสดงสถานะได้ด้วยแฟลก(flag) แฟลกเป็นรีจิสเตอร์ขนาด 8 บิท ซึ่งสามารถรวมกับแอคคิวมูเลเตอร์เป็นรีจิสเตอร์ขนาด 16 บิทได้ เราสามารถใช้คำสั่งในการเคลื่อนย้ายข้อมูลจากแอคคิวมูเลเตอร์ A และแฟลก F ไปเก็บไว้ใน A' และ F' ได้เพื่อ

ทำให้การใช้งานของ A และ F มีประสิทธิภาพดียิ่งขึ้น

หน่วยคำนวนทางคณิตศาสตร์และลอจิก (ALU-arithmetic and logic unit) การประมวลผลที่สำคัญของซีพียูของคอมพิวเตอร์ยังขึ้นอยู่กับหน่วยคำนวนทางคณิตศาสตร์และลอจิก(ALU) ซึ่งจะนำข้อมูลที่อาจมาจากภายนอกซีพียูหรือภายในซีพียูก็ได้มาประมวลผล การประมวลผลที่สำคัญจะประกอบด้วย

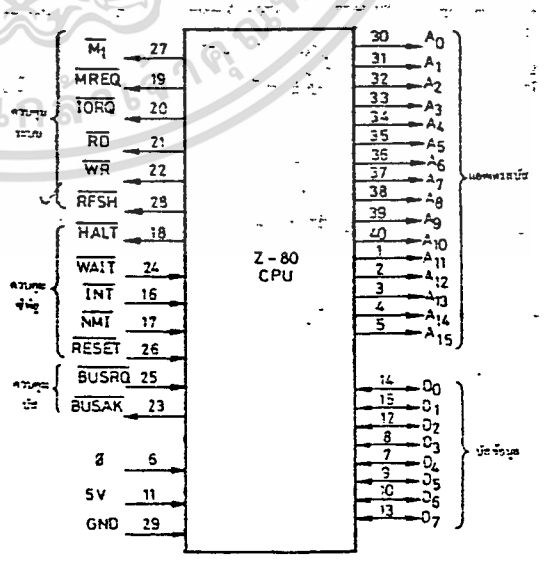


รูปที่ 2.3
แสดงการทำงานของ ALU ภายใน

รีจิสเตอร์คำสั่งและส่วนควบคุม (Instruction register and control) ในการกระทำการเฟชซีพียูจะอ่านคำสั่งจากหน่วยความจำที่เป็นส่วนของโปรแกรม โดยรอคำสั่งนั้นมาเก็บไว้ใน IR เพื่อทำการถอดรหัสคำสั่งและส่งสัญญาณควบคุมการทำงานภายในซีพียู หรือควบคุมการทำงานของระบบ สัญญาณควบคุมเหล่านี้จะออกมาในจังหวะต่างๆกัน เพื่อใช้ควบคุมในการทำงานต่อไป

การจัดขาของ Z-80

Z-80 ซีพียูเป็นไอซีไมโครโปรเซสเซอร์ที่มีขาเพียง20 ขา โดยหลักการแล้ว Z-80 เป็นซีพียูได้โดยสมบูรณ์กล่าวคือไม่ต้องประกอบกับอุปกรณ์ประกอบอื่นที่จะแยกการทำงานเพื่อจะรวมเป็นซีพียู ส่วนของสัญญาณจะประกอบด้วยบัสแอดเดรส บัสข้อมูลและสัญญาณควบคุม การจัดวางขาแสดงได้ดังรูปที่ 2.4



รายละเอียดของขาต่างๆแสดงได้ดังนี้

A0-A15 *บัสแอดเดรส* สัญญาณที่ออกมาจากขาไอซีเหล่านี้จะให้แอดดีฟขณะhighโดยขาเหล่านี้เป็นเอาท์พุทแบบไตรสเตท บัสแอดเดรสมีด้วยกันทั้งหมด 16 สายเพื่อให้ซีพียูติดต่อกับหน่วยความจำได้มากถึง2(16)=64kไบท์ นอกจากนี้ส่วนของแอดเดรสยังเป็นตัวกำหนดเบอร์พอร์ทของอุปกรณ์ อินพุท-เอาท์พุท โดยขณะที่ซีพียูกระทำคำสั่งเกี่ยวกับอินพุทหรือเอาท์พุทค่าของแอดเดรสบัสใน 8 บิทล่าง(A0-A7)จะแสดงค่าเบอร์พอร์ท ดังนั้นเราจึงมีอุปกรณ์อินพุทหรือเอาท์พุทได้ทั้งหมด 2(8) = 256 พอร์ท ในขณะช่วงเวลารีเฟชเมื่อสัญญาณปรากฏขึ้นที่ขารีเฟช (refb) ค่าในแอดเดรสบัสA0-A7จะแสดงค่าแอดเดรสของหน่วยความจำที่จะได้รับการรีเฟช

D0-D7 *บัสข้อมูล* (data bus) เป็นลักษณะของบัสแบบ2ทิศทางZ-80ซีพียูมีบัสข้อมูล 8เส้น บัสข้อมูลเป็นเส้นทางผ่านของข้อมูลระหว่างซีพียูกับหน่วยความจำ ซีพียูกับอุปกรณ์อินพุท-เอาท์พุทหรือการติดต่อระหว่างอุปกรณ์อินพุท-เอาท์พุทกับหน่วยความจำ

M1 Machine cycle one มีลักษณะเป็นแอคตีฟที่ลอจิก "0" M1บอกให้ทราบว่าซีพียูกำลังอยู่ในสภาวะเฟช ในขณะที่ซีพียูเฟชคำสั่งที่มีออฟโค้ดสองไบท์ส่วนของM1จะสร้างขึ้นขณะเฟชในแต่ละไบท์ ลักษณะของคำสั่งที่มีออฟโค้ดสองไบท์จะขึ้นต้นด้วย นอกจากนี้ M1 ยังสร้างสัญญาณร่วมกับ IQRQ เพื่อบอกสถานะการตอบรับการอินเตอรรัพท์

MREQ Memory request เป็นลักษณะไตรสเตท ให้ลอจิกแอคตีฟที่"0"เป็นสายสัญญาณที่บอกให้ทราบว่าซีพียูต้องการเขียนหรืออ่านหน่วยความจำตามแอดเดรสที่ปรากฏอยู่ในแอดเดรสบัส

IORQ Input output request เป็นเอาท์พุทลักษณะไตรสเตทให้ลอจิกแอคตีฟที่ "0" บอกให้ทราบว่าซีพียูต้องการติดต่อกับอุปกรณ์อินพุท-เอาท์พุท โดยแอดเดรสบัส8บิทล่างจะให้แสดงค่าเบอร์พอร์ท ส่วนบัสข้อมูลจะแสดงข้อมูลที่จะมีการส่งถ่ายระหว่างซีพียูกับI/Oนอกจากนี้IORQถ้าเกิดขึ้นพร้อมกับสัญญาณ M1 บอกถึงสถานะที่ซีพียูกำลังตอบสนองผลการอินเตอร์รัพท์ โดยส่วนของบัสข้อมูลจะมีการส่งผ่านเข้ามาด้วยค่าของอินเตอร์รัพท์เวคเตอร์

RD Memory read เป็นเอาท์พุทที่ไตรสเตทและแอคตีฟขณะลอจิก "0" RDเป็นตัวบอกว่าขณะนี้ซีพียูต้องการอ่านข้อมูลจากหน่วยความจำหรืออุปกรณ์I/O

WR Memory write เป็นขาเอาท์พุทแบบไตรสเตทและแอคตีฟขณะลอจิก"0"WRเป็นสัญญาณบอกว่าซีพียูต้องการเขียนข้อมูลโดยเขียนในตำแหน่งที่แอดเดรสบัสกำหนด อาจเป็นหน่วยความจำหรืออุปกรณ์I/Oก็ได้

RFSH Refresh เป็นขาอินพุทแอคตีฟขณะลอจิก"0"RFSHบอกให้ทราบว่าสัญญาณในแอดเดรสบัส ในส่วน A0-A6 เป็นแอดเดรสที่ใช้ในการรีเฟรชหน่วยความจำไดนามิกส์ ส่วนบิท A7 จะเป็น "0" ส่วนบิทA15-A8จะแสดงค่าของรีจิสเตอร์

HALT halt state เป็นเอาท์พุทที่แอคตีฟด้วยลอจิก"0" สัญญาณHALTจะแสดงเมื่อซีพียูได้กระทำคำสั่งHALTและจะหยุดรอจนกว่าจะมีการอินเตอร์รัพท์หรือรีเซท ขณะที่อยู่ในช่วงHALTซีพียูจะเสมือนกำลังกระทำคำสั่ง NOP(no operation) เพื่อให้เกิดไซเคิลในการทำงาน เพื่อส่งสัญญาณไปกระทำการรีเฟรชหน่วยความจำชนิดไดนามิกส์

WAIT Wait เป็นขาอินพุทจะแอคตีฟด้วยลอจิก"0"WAITเป็นตัวกำหนดแสดงเพื่อบอกซีพียูให้ซีพียูหยุดรอ ในกรณีที่อุปกรณ์ อินพุท-เอาท์พุท หรือหน่วยความจำไม่สามารถรับหรือส่งข้อมูลได้ทัน WAIT จะเป็นตัวทำให้ซีพียูซิงค์ได้พอดีกับอุปกรณ์ อินพุท-เอาท์พุท ที่ทำงานด้วยความเร็วช้าๆ

INT Interrupt request เป็นขาอินพุทแอคตีฟด้วยลอจิก "0" INT เป็นสัญญาณที่สร้างขึ้นมาจากอุปกรณ์อินพุทเอาท์พุท เพื่อต้องการที่จะอินเตอร์รัพท์ซีพียู ซีพียูจะทำการตรวจสอบสัญญาณนี้ทุกๆครั้งที่จบการกระทำแต่ละคำสั่ง การตอบสนองของตัวการอินเตอร์รัพท์สามารถควบคุมได้ด้วยซอฟท์แวร์ ด้วยการเซทค่าอินเตอร์รัพท์ฟลิบฟลอบ(IFF)การตอบสนองอินเตอร์รัฟท์จะเกิดขึ้นได้ยังต้องให้ BUSRQ ไม่แอคตีฟ เมื่อซีพียูตอบสนองต่อการอินเตอร์รัฟท์ ซีพียูจะสร้างสัญญาณตอบด้วยการสร้างสัญญาณIORQระหว่างช่องเวลา M1

NMI Nonmaskable interrupt เป็นขาอินพุทที่จะทริกบอกซีพียูในขณะขอบพัลซ์ขาลงการอินเตอร์รัพท์ด้วยวิธีนี้ซีพียูจะให้ความสำคัญสูงกว่า INT กล่าวคือมันจะตอบสนองและกระทำทันทีด้วยการเริ่มเอ็กซีคิ้ว คำสั่งในตำแหน่ง 066H โดยอัตโนมัติ การกระโดดไปกระทำในกรณีนี้ ซีพียูจะเก็บค่าโปรแกรมเคาวเตอร์เดิมไว้ในสแตค เพื่อจะได้กลับไปทำงานเดิมเมื่อเสร็จสิ้นการอินเตอร์รัพท์ได้

RESET Resetเป็นขาอินพุทที่แอคตีฟด้วยลอจิก"0"การรีเซทในกรณีนี้จะมีผลดังนี้

1.ค่าของPCจะมีค่าเป็น"0"

2.IFFจะได้รับการDisable

3.รีจิสเตอร์Iจะมีค่า00H

4.รีจิสเตอร์Rจะมีค่า00H

5.จะมีการเซทอินเตอร์รัพท์มาอยู่ที่โหมด0

ระหว่างการรีเซทสายแอดเดรสบัสและบัสข้อมูลจะได้รับการกระทำให้มีค่าอิมพีแดนซ์สูงเพื่อแยกออกจากซีพียู ส่วนสายสัญญาณควบคุมจะได้รับการทำให้เป็นสัญญาณที่ไม่แอคตีฟ การรีเฟรชจะไม่เกิดขึ้น

BUSRQ Bus reqoest เป็นขาอินพุทที่แอคตีฟด้วยลอจิก "0" BUSRQ เป็นสัญญาณที่ส่งบอกกับซีพียูเพื่อต้องการให้ซีพียูควบคุมบัส กล่าวคือต้องการให้ซีพียูทำให้บัสแอดเดรสและบัสข้อมูลอยู่ในสถานะอิมพีแดนซ์สูง คือต้องการแยกซีพียูออกจากบัสนั่นเอง

BUSAK Bus acknowledge เป็ยขาเอาท์พุทแอคตีฟขณะลอจิก"0"BUSAKเป็นสัญญาณตอบว่าซีพียูได้แยกตัวเองออกจากแอดเดรสบัสและบัสข้อมูลเรียบร้อยแล้ว

- Clock สัญญาณนาฬิกาที่จะป้อนเข้าระบบ

แฟลก

ในZ-80ซีพียูจะประกอบด้วยแฟลกจำนวน 6 แฟลก และมีบิทที่ไม่ได้แสดงเป็นแฟลกอีก 2 บิท รวมเป็น 8 บิทเพื่อประกอบเป็นรีจิสเตอร์ F ส่วนของแฟลกแต่ละบิทสามารถที่จะเซทหรือรีเซทตามการกระทำของคำสั่งที่ซีพียูกำลังทำงาน นอกนี้ซีพียูยังสามารถใช้ตรวจสอบแฟลกเพื่อกระทำเงื่อนไขต่างๆ

ลักษณะการใช้แฟลกจะใช้ตัวอักษรย่อแทนแฟลก ดังนี้

C = แฟลกตัวทด

N = แฟลกแสดงการบวกหรือลบ

P/V = แฟลกแสดงพาริตี้และโอเวอร์โฟลว์

H = แสดงแฟลกตัวทดช่วย

Z = แฟลกแสดงค่าศูนย์

S = แฟลกเครื่องหมาย

X = ไม่ได้ใช้

รายละเอียดของแฟลกแต่ละตัวเป็นดังนี้

1.แฟลกตัวทด (C:Carry flag) แฟลกตัวนี้เป็นบิทที่ใช้สำหรับทดจากข้อมูลในรีจิสเตอร์ A เช่นเมื่อมีการบวกข้อมูล8บิท ผลบวกอาจเลยเป็น9บิท บิทที่เกินเลยจะทดเข้าไปเก็บไว้ที่บิทCนี้ ในทำนองเดียวกัน ถ้าซีพียูกระทำค่าสั่งลบ และมีการขอยืมค่าของแฟลกตัวนี้ก็จะได้รับการเซทให้มีค่าเป็น"1"เช่นกัน .

2.แฟลกศูนย์ (Z:Zero flag) ในแฟลกบิทจะได้รับการเซทให้มีค่าเท่ากับ "1" ถ้าผลของการกระทำทำให้รีจิสเตอร์A มีค่า"1"นอกเหนือจากนี้มันจะกระทำการรีเซท

3.แฟลกเครื่องหมาย (S:Sign flag) แฟลกนี้มีประโยชน์ในการบอกการกระทำของซีพียู ว่าผลลัพท์ที่เกิดขึ้นมีค่าเป็นบวก หรือเป็นลบ ถ้าเครื่องหมายของตัวเลขเป็นลบคือบท D7=1 จะปรากฏค่า"1"ในบิทนี้

4.แฟลกพาริตี้/ค่าเกิน (parity/over flow flag) (p/v) แฟลกนี้ใช้สำหรับเป็นตัวบอกพาริตี้ของผลลัพท์ในแอคคิวมูเลเตอร์ เมื่อให้มีการกระทำทางลอจิก และยังใช้แสดงสภาวะของค่าที่เกินกำหนดใน 8 บิท

5.แฟลกตัวช่วย (H:half carry) แฟลกนี้จะเป็นบิทที่ทำหน้าที่เป็นตัวทดหรือตัวยืมของตัวเลข BCD

6.แฟลกการลบ (N:subtracct flag) เนื่องจากในการกระทำทางคณิตศาสตร์ของตัวเลข BCD เพื่อจะได้มีการรับรู้ในการปรับค่าเมื่อกระทำ DAA ได้ถูกต้อง แฟลกนี้จะเป็นตัวบอกว่าคำสั่งที่ถูกกระทำเป็นบวกหรือลบ โดยถ้าเป็นลบ แฟลกนี้จะได้รับการเซทให้มีค่าเป็น"1"

การอินเตอร์รัพท์แบบมาสเคเบิ้ล(maskable interrupt)

การอินเตอร์รัพท์ต้องผ่านเข้ามาทางขาINTของซีพียู เมื่อซีพียูได้รับสัญญาณนี้แล้ว จะตรวจสอบสถานะของตัวเองว่า จะตอบสนองต่อการอินเตอร์รัพท์หรือไม่ ซึ่งสามารถโปรแกรมได้ด้วยซอฟท์แวร์ ดังนั้นผู้โปรแกรมจึงสามารถกำหนดสภาวะการอินเตอร์รัพท์ ให้ได้รับการตอบสนองตรงส่วนใดของโปรแกรมได้ การอินเตอร์รัพท์ด้วยวิธีนี้แยกเป็น3โหมด คือโหมด 0 (IM0) โหมด 1 (IM1) โหมด 2 (IM2)

การอินเตอร์รัพท์โหมด 0

เมื่อมีการอินเตอร์รัพท์เกิดขึ้นและโปรแกรมทางซอฟท์แวร์ได้เซทโหมดการรัพท์อินเตอร์รัพท์เป็นโหมด 0 (IM0) และอีนาเบิ้ลการอินเตอร์รัพท์ไว้ การทำงานของซีพียูจะหยุดการเฟทช์คำสั่งถัดไป แต่จะตอบรับด้วยการส่งสัญญาณ M1 และ IORQ อ่านข้อมูล1ไบท์เข้ามาทางบัสข้อมูล ข้อมูลนี้ได้รับการส่งมาจากอุปกรณ์ I/O ที่อินเตอร์รัพท์มา เมื่อซีพียูอ่านข้อมูล จะถือว่าเป็นอ็อฟโค้ดทันทีและจะตีความหมายในการทำงาน คำสั่งขนาด1ไบท์ที่เหมาะในการใช้อินเตอร์คือ RST เมื่อซีพียูเอ็กซีคิ้วคำสั่ง RST ซีพียูจะเก็บข้อมูลเดิมไว้ที่สแตค แล้วเปลี่ยนค่าPCใหม่ตามลักษณะของการ RST นั้นๆ ดังนั้นซีพียูจะกระโดดข้ามไปทำงานตามที่ต้องการของการอินเตอร์รัพท์ได้

การตอบสนองต่อการอินเตอร์รัพท์ด้วยการส่งสัญญาณ M1 และ IORQทำการเฟทช์ข้อมูลจาก I/P มาเอ็กซีคิ้วนี้ อาจทำได้โดยการให้I/Pส่งคำสั่งอื่นที่ไม่ใช่คำสั่งRST(คำสั่งไบท์เดียว) แต่เป็นคำสั่งหลายไบท์

การใช้เวลาในการตอบสนองต่อการอินเตอร์รัพท์ ด้วยการส่ง M1 และ IORQ นี้ กระทำเหมือนในช่วงเวลาแมชีนไซเคิล M1 แต่ซีพียูจะใช้เวลานานกว่าโดยการเพิ่มTwขึ้น 2 ไซเคิลเพื่อให้เวลาแก่อุกรณ์ I/O ในการทำขบวนการเดซี่เชน(daisy chain) ในการจัดลำดับความสำคัญของการอินเตอร์รัพท์

โดยปกติการเซทให้อยู่ในโหมดนี้ ทำได้ด้วยคำสั่งทางซอฟท์แวร์คือ IM1 และการกำหนดให้รับการอินเตอร์รัพท์ได้หรือไม่ทำได้ด้วยคำสั่ง EI และ DI และเพื่อให้การทำงานเหมือนอยู่ในโหมดของ 8080 ทุกประการดังนั้นหลังจากการรีเซทซีพียู ซีพียูจะเซทตัวเองให้อยู่ในโหมด0โดยอัตโนมัติ

การอินเตอร์รัพท์โหมด 1

ในโหมดนี้สามารถกำหนดได้ด้วยคำสั่ง IM1 การอินเตอร์รัพท์กระทำเหมือนกับนอนมาสเคเบิ้ล ต่างเพียงการรีสตาร์ทมาที่ตำแหน่ง 38H(กรณีนอนมาสเคเบิ้ลไปกระทำที่ 66H) และจำนวนคาบเวลาที่ใช้ในโหมดหนึ่งนี้มีมากกว่าในนอนมาสเคเบิ้ล เพราะซีพียูต้องเพิ่มTwขึ้นอีก2สเตท การอินเตอร์รัพท์ในโหมดนี้สามารถดิสเอเบิ้ลได้ด้วยซอฟท์แวร์

การอินเตอร์รัพท์โหมด 2

ในโหมดนี้ทำให้Z-80มีขีดความสามารถเกี่ยวกับการอินเตอร์รัพท์สูงขึ้นมากการอินเตอร์รัพท์กำหนดด้วยคำสั่ง IM2 และการจะให้ซีพียูตอบสนองหรือไม่ด้วยคำสั่ง EI และ DI

การกระโดดไปยังโปรแกรมอื่นในขณะเมื่อซีพียูตอบสนองต่อการอินเตอร์รัพท์ ในกรณีนี้จะไปที่ใดก็ได้ โดยใช้แอดเดรสในการกระโดดนี้ได้ถึง16บิท ซึ่งทำได้สะดวกและรวดเร็วขึ้นอีกมาก

วิธีการตอบสนองต่อการอินเตอร์รัพท์นี้คือ เมื่อสัญญาณ INT เข้ามา และซีพียูตรวจสอบได้ในตอนสุดท้ายของคำสั่ง ซีพียูจะตอบสนองด้วยการส่ง M1 กับ IORQ ออกไป สัญญาณนี้จะเป็นตัวอุปกรณ์ที่ส่ง INT มาให้ส่งข้อมูลขนาด1ไบท์เข้าทางบัสข้อมูล ซึ่งข้อมูลนี้ถือว่าเป็นเวคเตอร์ของการอินเตอร์รัพท์ โดยข้อมูลในบิทD0จะต้องเป็น "0" ส่วนบิทอื่นจะเป็นอะไรก็ได้ ซีพียูจะนำเวคเตอร์นี้ไปเป็นข้อมูลแอดเดรสไบท์ที่มีนัยสำคัญต่ำ และข้อมูลจากรีจิสเตอร์Iภายในซีพียูเป็นข้อมูลไบท์ที่มีนัยสำคัญสูง เรียกหาไปยังข้อมูลในหน่วยความจำ 2 ไบท์ติดกันมานั้น มาโหลดใส่ PC หรือเป็นการอ้างแอดเดรสให้ PC แบบโดยทางอ้อมนั่นเอง

กรณีนี้จะเห็นว่าสามารถเซทค่าในรีจิสเตอร์Iให้เป็นอะไรก็ได้ และ I/O จะส่งข้อมูลเวคเตอร์มาประกอบรวมเพื่อบอกถึงค่าตารางในหน่วยความจำที่ต้องการ ด้วยวิธีการเช่นนี้จะทำให้การกระโดดไปยังโปรแกรมย่อยเกิดขึ้นที่ใดก็ได้

อุปกรณ์ไอซีที่ทำงานร่วมโดยใช้อินเตอร์รัพท์โหมดนี้มีหลายเบอร์ด้วยกัน เช่น Z-80PIO, Z-80CTC,Z-80SIO ฯลฯ ซึ่งสามารถส่งเวคเตอร์ให้กับซีพียูได้อย่างมีประสิทธิภาพ

การอีนาเบิ้ลและดิสเอเบิ้ลอินเตอร์รัพท์

สถานะภาพต่อการตอบสนองอินเตอร์รัพท์ ซีพียูจะทำการตรวจสอบที่ IFF หรือ อินเตอร์รัพท์ฟลิบฟลอบ ในกรณีของ Z-80 จะมีการฟลิบฟลอบอยู่ 2 บิท คือ IFF1และIFF2 โดยทั้ง 2บิทนี้จะได้รับการเกี่ยวข้องด้วยจากการกระทำของซีพียู โปรแกรมคำสั่งเข้ามาเซทรีเซทฟลิบฟลอบ

โดยหลักการIFF1ทำหน้าที่เป็นตัวอีนาเบิ้ลหรือดิสเอเบิ้ล การอินเตอร์รัพท์โดยที่IFF2จะมีหน้าที่หลักในการเก็บข้อมูลชั่วคราวของIFF1

ในขณะที่มีการรีเซทซีพียูทางฮาร์ดเซท ทั้ง IFF1และIFF2 จะได้รับการรีเซทไปด้วย การแสดงสถานะ "0" ของ IFF1 จะเป็นการดิสเอเบิ้ลการอินเตอร์รัพท์ กล่าวคือ IFF1="0" ซีพียูจะไม่รับรู้ต่อการอินเตอร์รัพท์ที่เข้ามาทาง INT การเซทIFFสามารถกระทำได้ด้วยคำสั่ง EI โดยสถานะภาพของ IFF1 และ IFF2 ที่จะเปลี่ยนแปลงเนื่องจากการกระทำต่างๆ

การกระทำ	IFF1	IFF2	
รีเซทซีพียู	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	IFF2 ⟶ แฟลกพาริตี้
LD A, R	•	•	IFF2 ⟶ แฟลกพาริตี้
เมื่อกระทำ NMI	0	•	
RETN	IFF2	•	IFF2 ⟶ IFF1
เมื่อกระทำ INT	0	0	
RETI	•	•	

ตาราง IFF

จากตารางพอสรุปได้ว่า การกำหนดอีนาเบิ้ลจะต้องทำการเซทฟลิบฟลอบ IFF1 หรือกล่างอีกนัยหนึ่ง การยอมให้เกิดการอินเตอร์รัพท์ได้ก็ต่อเมื่อซีพียูตรวจสอบ IFF1ว่าอยู่ในสภาวะอีนาเบิ้ลหรือไม่

การตรวจสอบนี้ในบางกรณีทำได้โดย การตรวจสอบทางบิทพาริตี้ คือการกระทำคำสั่ง LD A,I และ LD A,R จะมีผลทำให้ค่าของIFF2ไปเก็บยังพาริตี้แฟลก

เมื่อมีการอินเตอร์รัพท์แบบNMIขึ้นจะเกิดสภาวะดิสเอเบิ้ลทันทีที่ IFF1 กล่าวคือจะได้รับการรีเซท นั่นคือระหว่างการอินเตอร์รัพท์แบบNMIนี้ การอินเตอร์รัพท์แบบอื่นจะเข้ามาอีกไม่ได้ ซีพียูจะไม่รับรู้ทั้งสิ้น สถานะเดิมก่อนการอินเตอร์รัพท์แบบNMI(สถานะดิสเอเบิ้ล หรืออีนาเบิ้ล) จะได้รับการเก็บรักษาไว้ที่ IFF2 ซึ่งระหว่างนี้จะได้รับการตรวจสอบเช่นกันว่า ก่อนการเข้าไปสู่โหมดNMIสภาวะการเป็นอย่างไร และเมื่อกลับเข้าโปรแกรมหลักด้วยคำสั่ง RETN จะทำให้สถานะเดิมเก็บรักษาไว้ใน IFF1 ใหม่

การตอบสนองต่อINTก็จะทำให้IFF1และIFF2ได้รับการรีเซทเช่นกัน ดังนั้นเมื่อมีการ INT สัญญาณINTครั้งต่อไปจะไม่สามารถได้รับการตอบสนอง จนกว่าจะมีคำสั่งEI INT ส่วนการเอ็กซีคิ้วคำสั่ง RETI จะไม่มีผลทำให้ IFF1 และ IFF2 เกิดการเปลี่ยนแปลง

รายละเอียดการทำงานของ MM 58167

REAL TIME COUNTER

REAL TIME COUNTER เป็นตัวนับและจัดการเกี่ยวกับเวลา ถูกแบ่งเป็นDIGIT ละ 4 บิท ซึ่งการเข้าถึง REAL TIME COUNTER จะกระทำครั้งละ 2 DIGIT (ในขณะ READ และ WRITE ซึ่งแต่ละ DIGIT จะให้ค่า BCD ดังแสดงในตารางที่ 1 บิทที่ไม่ใช้จะถูก HOLD ด้วย LOGIT 0 ซึ่งเราไม่ต้องสนใจในขณะที่ทำการเขียนข้อมูลลงบน DATA BUS เหตุที่บางบิทไม่ใช้ก็เนื่องจากว่า ไม่จำเป็นต้องใช้ในการให้ข้อมูลแบบ BCD ของบางหลัก ตัวอย่างเช่นในหลักสิบของชั่วโมงจะไม่เกินเลข 2 ฉะนั้นเราจะใช้เพียง 2 บิทเท่านั้น ไม่ต้องใช้ในบิทที่ 6 และบิท 7 (ดูตารางที่ 1)

COUNTER ADDRESS	UNIT D_0	D_1	D_2	D_3	MAX BCD CODE	TENS D_4	D_5	D_6	D_7	MAX BCD CODE
1/10,000 OF SEC(00H)	-	-	-	-		D4	D5	D6	D7	9
1/10 & 1/100 (01H)	D0	D1	D2	D3	9	D4	D5	D6	D7	9
SECOND (02H)	D0	D1	D2	D3	9	D4	D5	D6	-	5
MINUTE (03H)	D0	D1	D2	D3	9	D4	D5	D6	-	5
HOURS (04H)	D0	D1	D2	D3	9	D4	D5	-	-	2
DAY OF WEEK (05H)	D0	D1	D2	-	7	-	-	-	-	0
DAY OF MONTH (06H)	D0	D1	D2	D3	9	D4	D5	-	-	3
MONTH (07H)	D0	D1	D2	D3	9	D4	-	-	-	1

ตารางที่ 1

RAM

MM 58167 มี RAMขนาด 56 บิทซึ่งใช้ในการเก็บข้อมูลเมื่อไฟตก หรือใช้เก็บข้อมูลการตั้งปลุกเพื่อที่จะเปรียบเทียบ (compare)กับ REAL TIME COUNTER ข้อมูลใน RAMจะสามารถเปรียบเทียบกับ REAL TIME COUNTER และมี DIGIT ที่ไม่ใช้คือ หลักหน่วยของ SEC., และหลักสิบของวันในสัปดาห์(เพราะไม่ใช้ใน REAL TIME COUNTE) ดูตารางที่ 1 ประกอบด้วย

RAM จะถูกกำหนดให้มีรูปแบบเหมือนกับตัวนับเวลา อย่างไรก็ตามยังมีบิทที่ยังไม่ได้ใช้อยู่ ซึ่งบิทที่ยังไม่ได้ใช้ในตัวนับเวลานี้จะเปรียบเทียบกับ "0" ใน RAM

INTERUPT และตัวเปรียบเทียบ

มีสัญญาณ interupt อยู่ 2 อย่าง อย่างแรกคือ Interupt output (Active "1") เอาท์พุทนี้สามารถจะโปรแกรมให้เกิดสัญญาณทางออกได้ 8 อย่าง คือ 10Hz,1Hz นาที/ครั้ง, 1 ชั่วโมง/ครั้ง,1 วัน/ครั้ง,1 สัปดาห์/ครั้ง,1 เดือน/ครั้ง และเมื่อรวมแรมเข้ากับตัวนับเวลาเกิดการเปรียบเทียบขึ้น

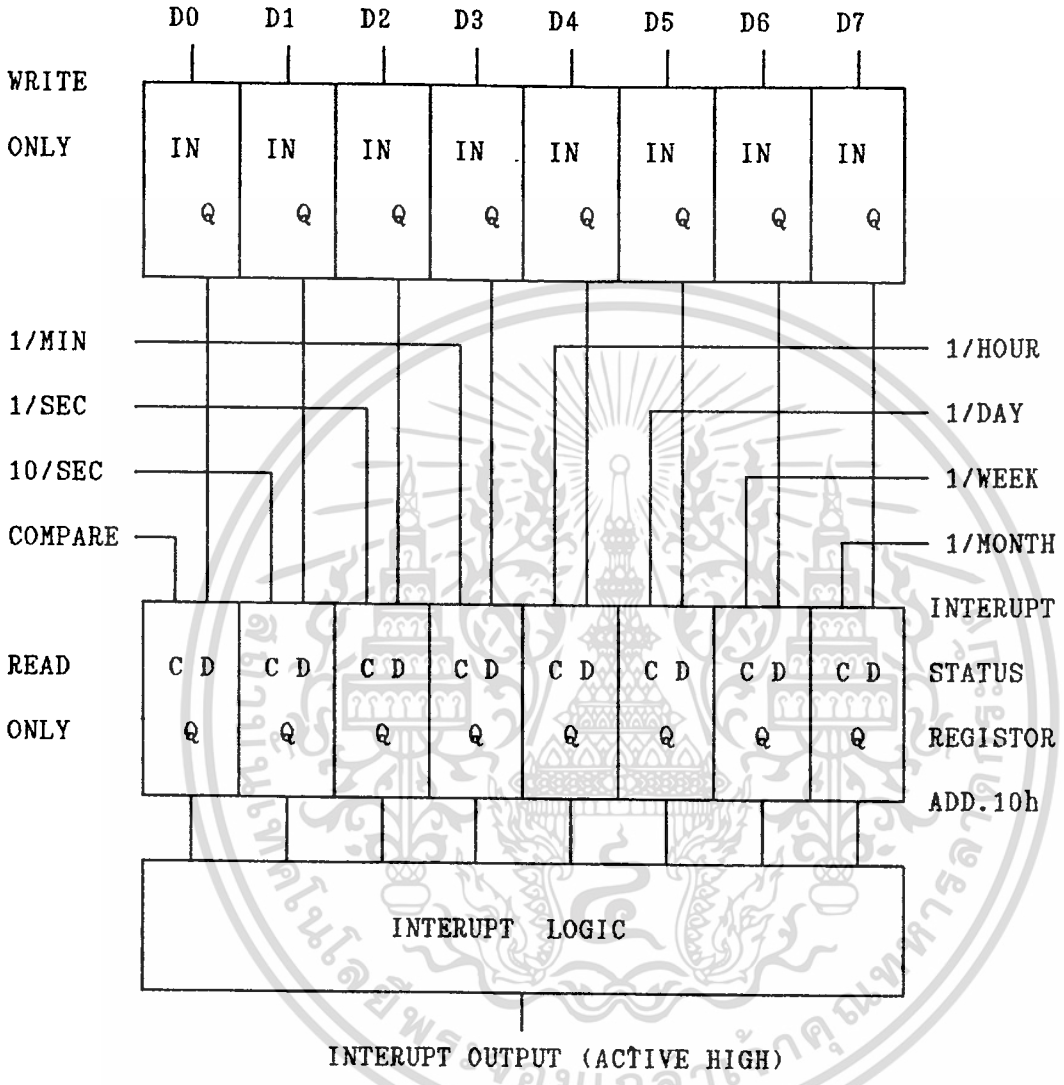
วิธีการที่จะอีนาเบิลสัญญาณอินเตอร์รัพต์คือ ให้ลอจิก"1"แก่รีจีสเตอร์ควบคุมการอินเตอร์รัพต์ ในบิตที่ตรงกับความถี่ที่เราต้องการจะให้เกิดสัญญาณอินเตอร์รัพต์

วิธีการที่จะ Enable สัญญาณ Interupt คือให้ลอจิก "1" แก่ Interupt control registor ในบิทที่ตรงกับความถี่ที่เราต้องการจะให้เกิดสัญญาณ Interupt ดูรูปที่ 2 ประกอบ เช่น ต้องการให้เกิดสัญญาณ Interupt ทุกๆ 1 นาทีก็ให้ D2 เป็น "1" เขียนไปที่ Interupt control registor ดูได้จากตารางที่ 2

เราสามารถเซท Interupt control registor ครั้งละ บิท หรือมากกว่าก็ได้ ตัวอย่างเช่น เราต้องการให้มีสัญญาณ Interupt ทุกวินาที และ ทุกๆ ชั่วโมงก็เซทบิทที่ 3 (D3) กับบิทที่ 2 (D2) ในที่นี้คือ OCH โดยเขียนไปที่แอดเดรสของ Interupt control registor (ดูรูปที่ 2)

เมื่อเวลานับมาถึงค่าสูงสุดของแต่ละภาคจะทำให้เกิด CLOCK กับ Interupt control registor ซึ่งทำให้ Interupt output เป็น HIGH การอ่าน Interupt status registor นี้จะได้ของมูลบน DATA BUS ซึ่งประกอบด้วยบิทที่ทำให้เกิดการอินเตอร์รัพท์โดยจะให้ค่าเป็น "1" ที่บิทนั้น

INTERUPT CONTROL REGISTOR



อินเตอร์รัพท์อีกอย่างหนึ่งคือ Stanby Interupt (Open drain output ,Active low) อินเตอร์รัพท์ตัวนี้จะเกิดขึ้นเมื่อเราได้ทำการ Enable ไว้และเกิดการเปรียบเทียบ (Compare) ใน RAM กับ Real time counter การ Enable ทำได้โดยการเขียน 01H ไปที่ Address 16H จะเป็นการ Disable

POWER DOWN MODE

ขา Power down เป็นขา Ship select ที่สำคัญตัวที่สอง มันจะ Disable สัญญาณทั้งหมด ยกเว้นสัญญาณ Stanby Interupt เมื่อขา Power Down ได้รับ Logic Low MM58167 จะไม่ตอบสนองสัญญาณแก่ภายนอกแต่นาฬิกาจะยังคงเดินตามปกติ และจะยังให้สัญญาณ Stanby Interupt (ขา 14) ถ้าได้มีการโปรแกรมขานี้ให้ทำงานไว้ก่อนแล้ว

เมื่อต้องการเปลี่ยนโหมดการทำงานปกติมาเป็น Stanby Mode ควรจะให้ขา Power Down เป็นลอจิก "0" อย่างน้อย 1μS ก่อนที่จะทำการลดระดับลงมาเป็น Stanby Mode เมื่อการเปลี่ยนกับมาสู่ก เรทำงานปกติผู้ใช้ต้องมั่นใจว่าขาอินพุทอื่นๆ ต้องเป็นสัญญาณที่ถูกต้องก่อนที่จะกลับมาสู่โหมดทำการปกติ ทั้งนี้เพื่อป้องกันข้อมูลของนาฬิกาเสียไป จะทำให้นาฬิกาเดินผิด ได้แก่ขา CS,RD,WR ของ MM58167 มีสัญญาณเปลี่ยนแปลงในขณะที่กลับสู่โหมดปกติ จะทำให้การเขียนข้อมูลไปที่ Real Time Counter หรือใน RAM

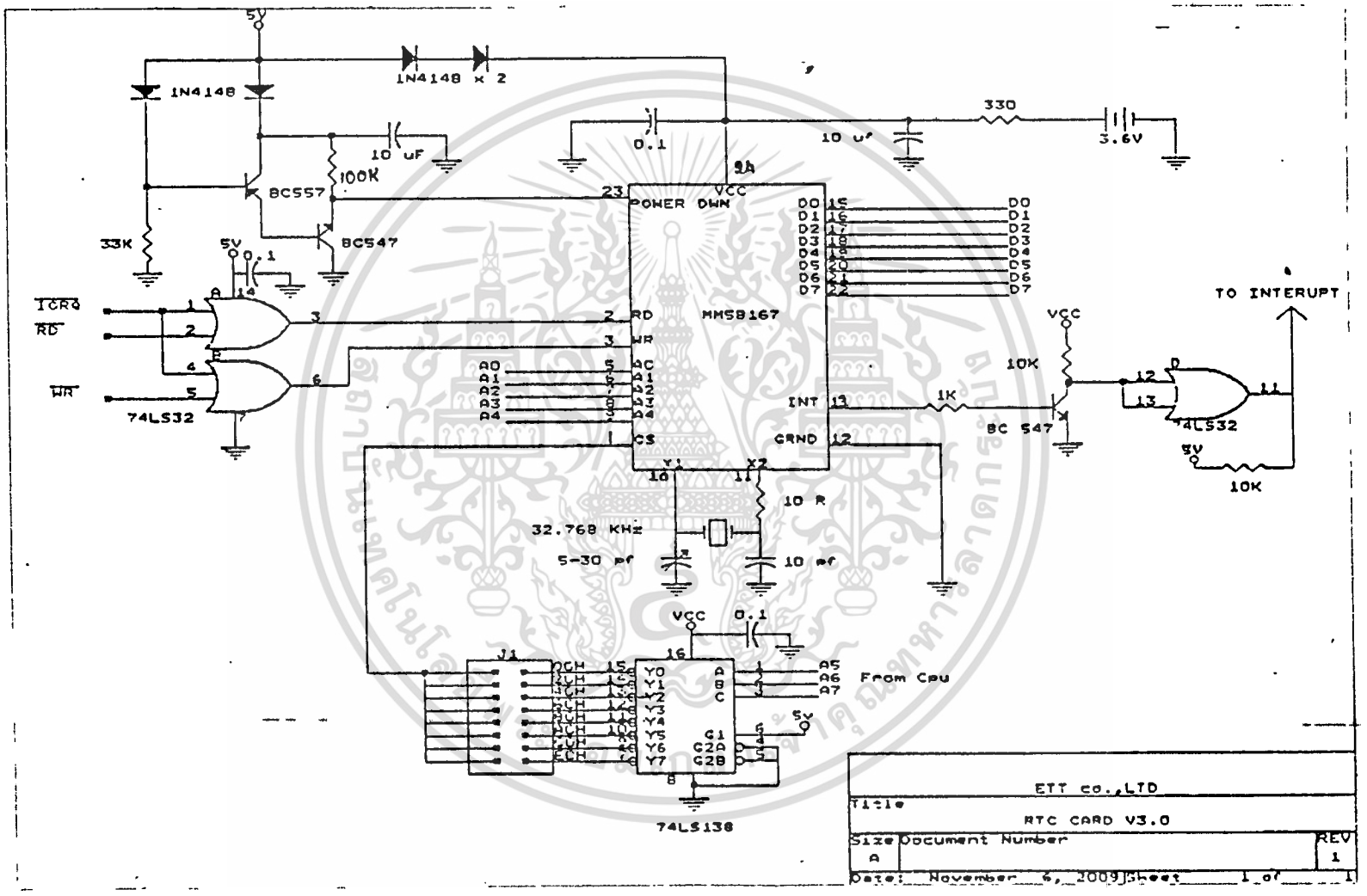
COUNTER AND RAM RESET ; GO COMMAND

ตัวนับเวลา (Counter) และ RAM สามารถรีเซสได้โดยเขียน OFFH ที่ Address 12H,13H ตามลำดับการให้ Pulse ของการเขียนไปที่ Address 15H (GO COMMAND) จะรีเซทตัวนับของวินาที ขณะทำการเขียนไปที่ Address 15H นี้ MM58167 จะไม่สนใจข้อมูลบน DATA BUS ผลของคำสั่งมีดังนี้

ถ้าตัวนับของวินาทีได้ค่ามากกว่า 39 เมื่อเราใช้คำสั่ง GO (Add 15H) จะทำให้หลักของนาทีเพิ่มขึ้น ในกรณีอื่นๆจะไม่มีผลต่อหลักนาที

STATUS BIT

STATUS BIT จะบอกผู้ใช้ว่าขณะที่ทำการอ่านตัวนับ(Counter)นั้น ตัวนับอยู่ในช่วงของการ UPDATE เวลาของมูลที่อ่นได้อาจมีการผิดพลาดเกิดขึ้น STATUS BIT นี้จะอ่าได้จาก Address 14H ของ RTC โดยจะให้ลอจิก "1" ที่บิท 0 ของ DATA BUS ในขณะที่บิทอื่นๆ เป็น 0 หมด ถ้าสัญญาณนี้ปรากฏขึ้นภายหลังการอ่านตัวนับควรมีการอ่านตัวนับใหม่ที่ขอบขาลงของสัญญาณ READ ที่ Adrress 14H รีเซท STATUS BIT ด้วย


5V
1N4148
1N4148 x 2
10 uF
100K
BC557
BC547
33K
0.1
330
3.6V
10 uF
VCC
POWER DWN
MM58167
RD
WR
CS
INT
GRND
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
A4
ICR3
RD
WR
74LS32
TO INTERUPT
10K
1K
BC 547
74LS532
32.768 KHz
5-30 pf
10 pf
10 R
J1
From Cpu
74LS138
ETT co.,LTD
RTC CARD V3.0
Size A
Document Number
REV 1
Date: November 6, 2009 Sheet 1 of 1

จออักษร LCD สำหรับชิงเกิ้ลบอร์ด

LCD โมดูลชนิดด็อตแมกริกว์นี้ออกได้เป็นพวกๆคือ

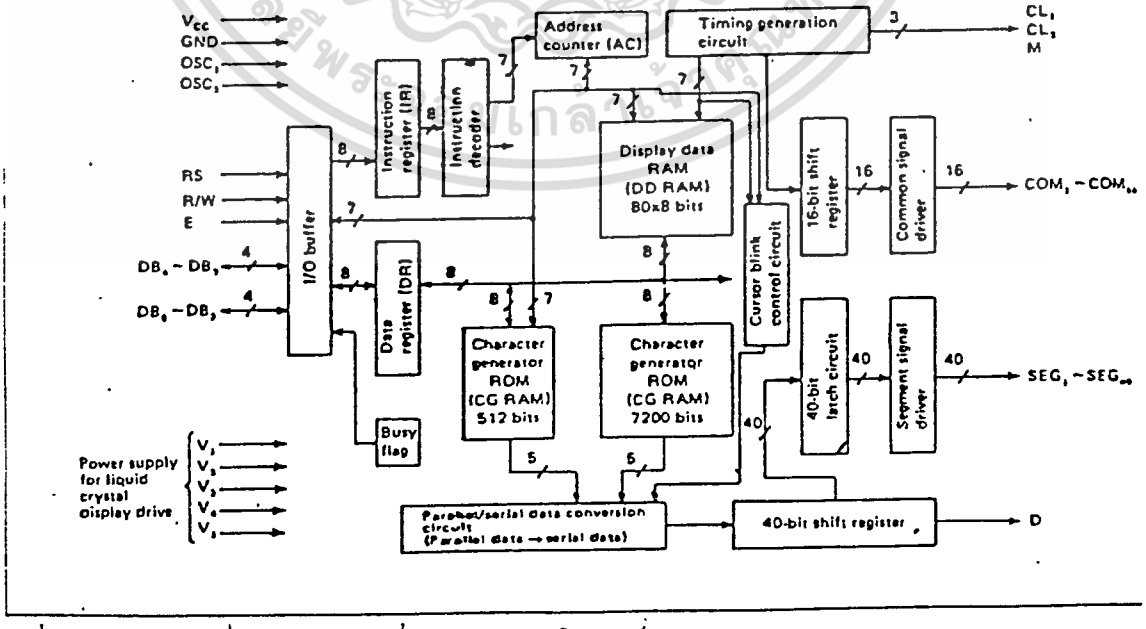
- charcter LCD module
- graphic LCD module
- segment display type LCD module

โดยในแต่ละแบบนี้ก็จะมีส่วนประกอบใหญ่ๆ แบ่งได้เป็น

จอ LCD แบบด็อตแมตริกซ์(dot matrix LCD) เป็นตัวแสดงผลให้เรามองเห็นในลักษณะการปิดและเปิดตัวเองกับแสงวก็คือ ส่วนของที่เป็นตัวกระจกบรรจุผลึก

ไดรเวอร์(driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับผลึก LCD อีกหนึ่งโดยมีเบอร์ที่นิยมใช้ใน LCD โมดูล เช่น HD4410H,MSM5259

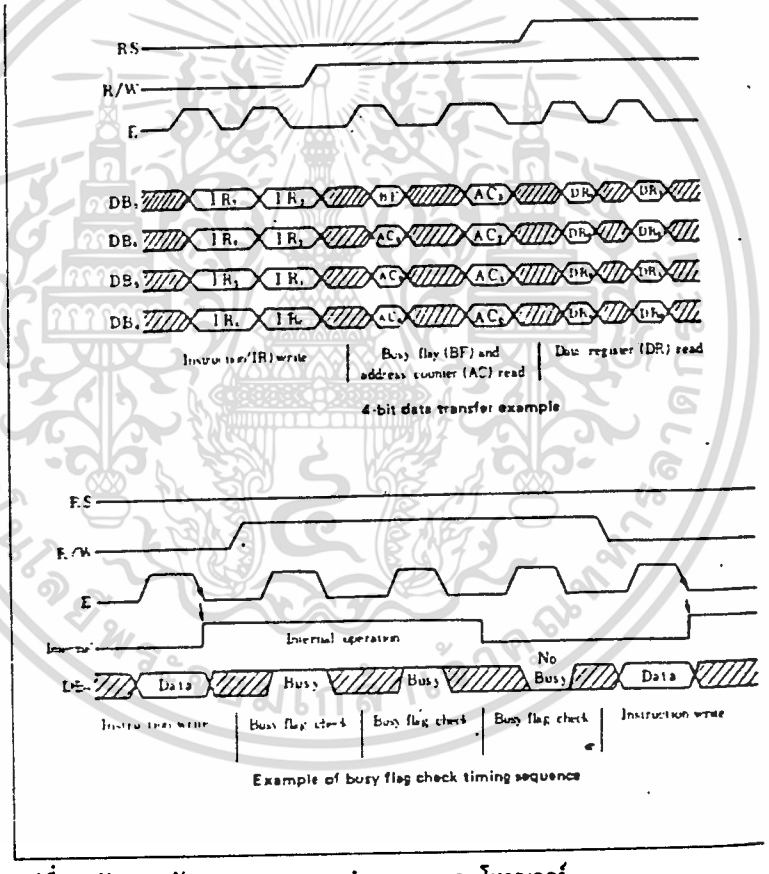
คอนโทรลเลอร์(controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาและจัดการควบคุม LCD โมดูลให้ทำงานแสดงผลต่างๆ เช่น การลบจอภาพ,การเกิดตัวอักษร,เป็นต้น โดยมีเบอร์ ไอซีที่นิยมกนคือ HD44780 ซึ่งใช้ในแบบ character LCD module คือการแสดงเป็นตัวอักษรโดดๆ เป็นส่วนใหญ่ ส่วน HD61830 จะใช้ในแบบ graphic LCD module หรือแสดงในลักษณะกราฟิกได้



รูปที่ 2 โครงสร้างวงจรภายในของ HD44780 ซึ่งเป็น LCD คอนโทรลเลอร์

ในการศึกษาการทำงานและใช้งาน LCDโมดูล นั้นไม่ใช่เรื่องยากเลยถ้าเราสามารถทำความเข้าใจในส่วนของคอนโทรอนเลอร์ได้เพียงหอแล้ว และโดยมาก LCD โมดูลในแต่ละบริษัทมักใช้ตัวคอนโทรลเลอร์ที่มีหลักการทำงานเหมือนกันเป็นส่วนใหญ่ใน LCD โมดูล แต่ละขนาดจำนวนตัวอักษรหรือจำนวนบรรทัดก็มีหลักการทำงานแบบเดียวกันทั้งหมดไอซีที่นิยมมากที่สุดตัวหนึ่งที่เป็น LCD คอรโทรลเลอร์ก็คือ เบอร์ HD44780 โดยรูปแบบการทำงานของมันได้เป็นมาตรฐานให้กับ LCD คอรโทรลเลอร์ตัวอื่นๆด้วย

HD44780 เป็นไอซี LSI ตัวหนึ่งใช้ควบคุม LCD โโยแสดงผลในรูปตัวอักษรหรือสัญญาณต่างๆ ตัวมันเอง เมื่อเริ่ป้อนไฟให้ HD44780 ก็จะมีการรีเซทตัวมันเองโดยใช้เวลาประมาณ 10ms หลังจากไฟ Vdd ถึง 4.5 โวลต์แล้ว หลังจากนั้นจะเซทตัวเองดังนี้:



รูปที่ 3 ลักษณะสัญญาณควบคุมการทำงานของคอนโทรลเลอร์

1. DISPLAY CLEAR ทำการลบข้อมูลจอภาพ LCD
2. FUNCTION SET ทำงานการเซทคุ่ภายใน

DL = 1 :เป็นการเซทให้การติดต่อเป็นแบบ 8 บิต

N = 0 :เซทเป็น 1 บรรทัดการแสดงผล

f = 0 :จำนวนจุดต่อหนึ่งต่ออักษรเท่ากับ 5*7

3. DISPLAY ON/OFF

D = 0 :display off

c = 0 :cursor off

b = 0 :bink off

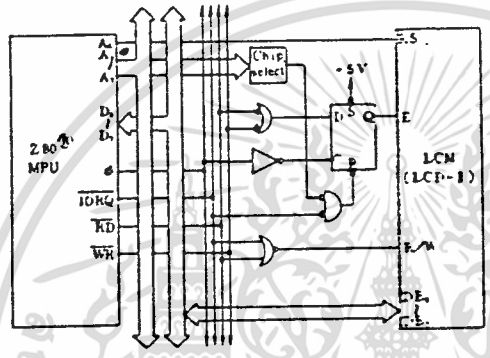
4. ENTRY MODE SET

I/D = 1 :+1(เพิ่มค่าเคาน์เตอร์ขึ้น 1)

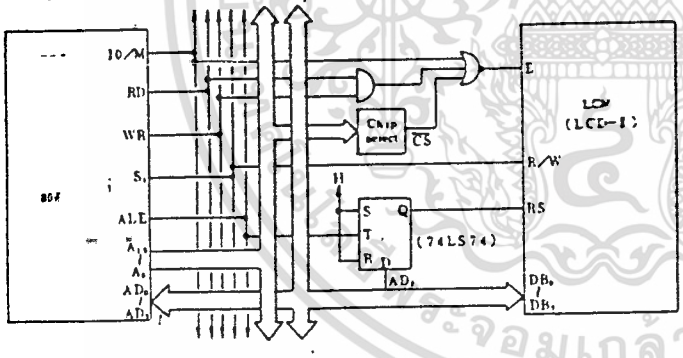
s = 0 :no shift

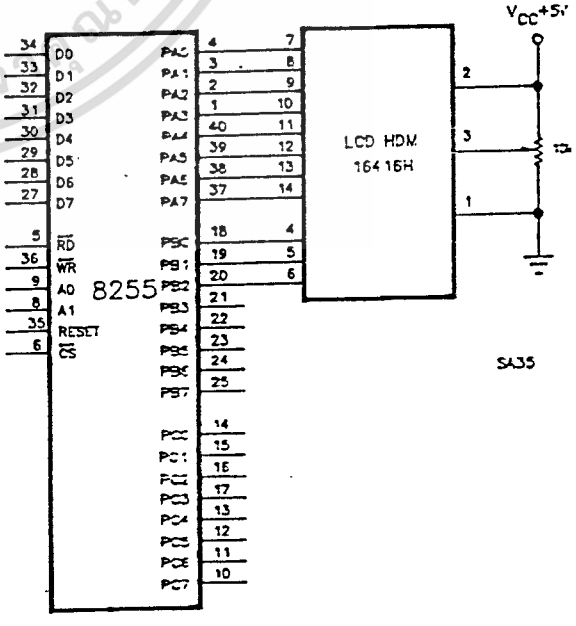
เมื่อเริ่มเปิดเครื่องทำงานแล้วก็จะต้องส่งคำสั่งควบคุมให้มันเริ่มทำงานดังขั้นตอนในรูปที่ 7

รูปที่ 4. ตัวอย่างการต่อ LCD โมดูลกับไมโครโปรเซสเซอร์ Z80



รูปที่ 5 การต่อ LCD โมดูลในลักษณะเป็นหน่วยความจำอันหนึ่งที่เข้าถึงได้





8255 FROM ET BOARD V3.0

รูปที่ 6 ลักษณะการต่อใช้งาน LCD โมดูลกับพอร์ตของอีทีบอร์ด

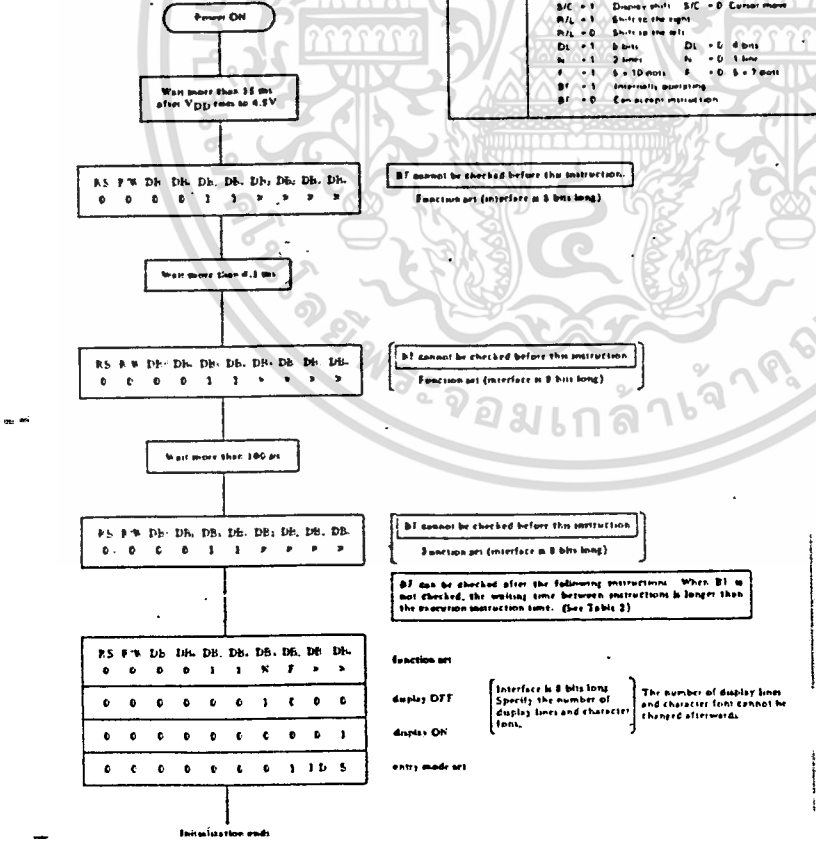
ตารางที่ 1 สรุปคำสั่งของ HD44780

Table 2 Instructions

Instruction	Code RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution time (fosc or fcp = 250 kHz) Note 1	Execution time (fosc or fcp = 270 kHz) Note 2
Clear display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	82 μs × 1.64 ms	82 μs × 4.8 ms
Return home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	40 μs × 1.6 ms	120 μs × 4.8 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 μs	120 μs
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40 μs	120 μs
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents.	40 μs	120 μs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display lines (L) and character font (F).	40 μs	120 μs
Set CG RAM address	0	0	0	1	ACG						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs	120 μs
Set DD RAM address	0	0	1	ADD							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs	120 μs
Read busy flag & address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	1 μs	1 μs
Write data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs	120 μs
Read data to CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs	120 μs
	I/D = 1: Increment (+1) I/D = 0: Decrement (−1) S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5 × 10 dots F = 0: 5 × 7 dots BF = 1: Internally operating BF = 0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM ACG: CG RAM address ADD: DD RAM address. Corresponds to cursor address. AC: Address counter used for both of DD and CG RAM address.	Execution time changes when frequency changes Example: When fosc is 270 kHz 40 μs × $\frac{250}{270}$ = 37 μs	

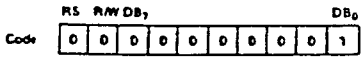
*No effect
Notes 1 Applied to models driven by 1/8 duty or 1/11 duty.
2 Applied to models driven by 1/16 duty.

รูปที่ 7 ขั้นตอนการรับคำสั่งควบคุมให้ LCD โมดูลทำงาน



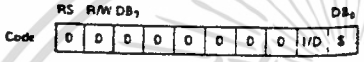
รายละเอียดของคำสั่ง HD44780

clear display



คำสั่งนี้เป็นการเขียนช่องว่างหรือ space(ASCII 20H) เข้าไปใน DD RAM (display data RAM) ทั้งหมด และทำการเซท DD RAM แอดรสให้เป็นศูนย์ตัวเคอร์เซอร์จะกลับไปอยู่ในตำแหน่งมุมบนซ้ายมือของจอภาพ เซทI/D= 1,S ไม่มีการเปลี่ยน

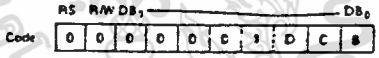
ENTRY MODE SET



BIT I/D: โดยจะเป็นตัวกำหนดให้ว่าเมื่อเขียนหรืออ่านข้อมูลแล้วจะทำให้ DD RAM แอดรสเพิ่มขึ้นหรือลดลงหนึ่งโดย 1 = เพิ่ม, 0 = ลดลงหนึ่ง

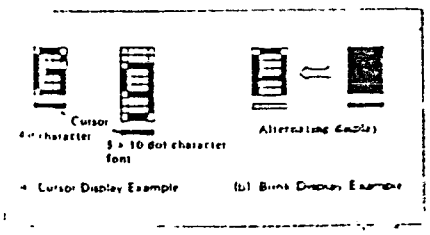
BIT S: เป็นตัวกำหนดการแสดงผล ถ้า s = 1 จะเป็นการใส่ข้อมูลแล้วตัวเคอร์เซอร์อยู่กับที่ ข้อมูลถูกดันไปทางซ้าย ถ้า S = 0 ข้อมูลจะอยู่ที่ตัวเคอร์เซอร์จะถูกดันไปขวามือ

DISPLAY ON/OFF CONTROL

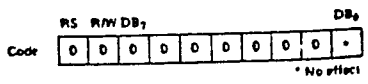


BIT D :เป็นบิตที่ใช้ควบคุมการเปิดหน้าจอ โดยถ้า D + 1 จะ ON และ D = 0 จะ OFF

BIT C :ใช้แสดงเคอร์เซอร์เมื่อให้บิต C = 1 และถ้าไม่ต้องการแสดงเคอร์เซอร์ก็ให้บิต C = 0 โดยตัวเคอร์เซอร์จะอยู่ที่แถวที่ 8 ในแบบ 5*7 จุดและจะอยู่แถวที่11 ในแบบ 5*10 จุด BIT B :เป็นบิตที่ใช้เซทการกะพริบของเคอร์เซอร์ โดย B = 1 มีการกะพริบ B = 0 ไม่มีการกะพริบ มีระยะเวลาการกะพริบประมาณ 379.2ms

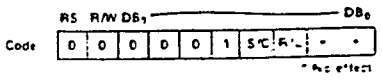


RETURN HOME



คำสั่งนี้จะทำการเซท DD RAM แอดรสให้เป็นศูนย์ ตัวเคอร์เซอร์จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพข้อมูลในจอภาพไม่เปลี่ยน

CURSOR OR DISPLAY SHIFT



เป็นคำสั่งกำหนดให้ตำแหน่งเคอร์เซอร์หรือข้อมูลไปปรากฏทางซ้ายหรือขวาโดยไม่ต้องใช้คำสั่งเขียนหรืออ่านโดย

S/C R/L

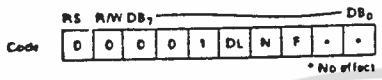
0 0 ทำการย้ายเคอร์เซอร์ไปจากตำแหน่งเดิมไปซ้ายมือ 1 ตำแหน่ง

0 1 ทำการย้ายเคอร์เซอร์ไปจากตำแหน่งเดิมไปขวามือ 1 ตำแหน่ง

1 0 เป็นการดันตัวอักษรที่ปรากฏไปทางซ้าย

1 1 เป็นการดันตัวอักษรที่ปรากฏไปทางขวามือ

FUNCTION SET



BIT DL: เป็นการเซทการติดต่อว่าจะให้เป็นแบบ 8 บิต หรือ 4 บิต โดยถ้าต้องการติดต่อ 4 บิต DL = 0 และ 8 บิต DL = 1

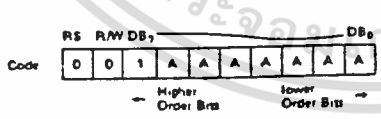
N: เป็นการเซตบรรทัดการแสดงผล N = 0 แสดง 1 บรรทัด N = 1 แสดง 2 บรรทัด ในกรณีมากกว่า 2 บรรทัด ก็เซต N = 1

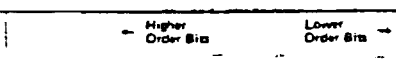
F: เป็นการเซตขนาดจุดของการแสดงผลเป็นแบบ 5*7 หรือ 5*10 โย F = 0 เป็นแบบ 5*7 และ F = 1 เป็นแบบ 5*10

N F	No. of display lines	Character font	Duty factor	Remarks
0 0	1	5 × 7 dots	1/8	
0 1	1	5 × 10 dots	1/11	
1 *	2	5 × 7 dots	1/16	Cannot display 2 lines with 5 × 10 dot character font

No effect

SET CG RAM ADDRESS





แหน่ง บิต 5 บิต 0 และเมื่อกำหนดแอดเดรสแล้วก็จะทำให้การเขียนข้อมูลลงใน CG RAM โดยเป็นลักษณะบิตต่อบิตบนจอ 1 ตัวอักษร คือขนาด 5*7 นั้นจะใช้ข้มูล บิต 4 ถึงบิต 0 ต่อ 1 ไบต์เท่านั้น ซึ่ง 1 ตัวอักษรจะใช้ข้อมูล 8 ไบต์ด้วยกัน ให้ดูตารางประกอบไปด้วย และเมื่อเขียนข้อมูลลงใน CG RAM แล้วเวลาจะใช้งานก็ให้เขียนข้อมูลใน DD RAM คือ ข้อมูลตำแหน่งในตาราง character ที่ตำแหน่ง 00H-07H

ตัวอย่างโปรแกรมการเขียนข้อมูลตัวหนังสือภาษาไทยเข้าไปใน CG RAM ตำแหน่งที่ 00H,01Hและ 02H และนำมาแสดงผลทางจอ LCD โดยใช้ 2 บรรทัดในการแสดงผล

การใช้งาน LCD โมดูลนั้นที่สำคัญคือ ต้องเข้าใจในตัวคอนโทรลเลอร์ของ LCD โมดูลนั้น โดยโทรลเลอร์ทุกๆ บริษัทจะมีการทำงานที่เหมือนกันเป็นส่วนใหญ่

Table 3 The relation between the operation and the combination of RS, R/W

RS	RW	E	OPERATION
0	0		Write instruction code
0	1		Read busy flag and address counter
1	0		Write data
1	1		Read data

When performing data and instruction code by 4 bit, transfer RS, R/W every time.

รูปที่ 8 แสดงความสัมพันธ์ระหว่างสัมพันธ์ RS และ R/W

ตารางที่ 4 รูปแบบอักษรที่สามารถแสดงได้

CHARACTER FONT TABLE

NOTE: CGRAM is a CHARACTER GENERATOR RAM having a storage function of character pattern which enable to change freely by user's program.

ขาต่างๆ ในการต่อใช้งาน HD44780

1. RS (register selection) จะเป็นขาเลือกรีจิสเตอร์ภายในซึ่งมีอยู่ 2 ตัวคือ instruction registor (IR) และ data register (DR) โดยถ้าเป็น 1 จะเป็นการเลือกข้อมูลและถ้าเป็น 0 จะเป็นการเลือกคำสั่ง

2. R/W(read/write) เป็นตัวเลือกว่าจะเขียนหรือจะอ่านข้อมูลจากตัวไอซีโดยอ่านข้อมูล = 1,เขียนข้อมูล = 0

3. E (enable signal) เป็นขากำหนดสภาพการรับเขียนอ่านข้อมูล

4. DB0-DB7 เป็นขารับส่งข้อมูลจากไอซี

5. VDD ไฟเลี้ยงตัววงจร

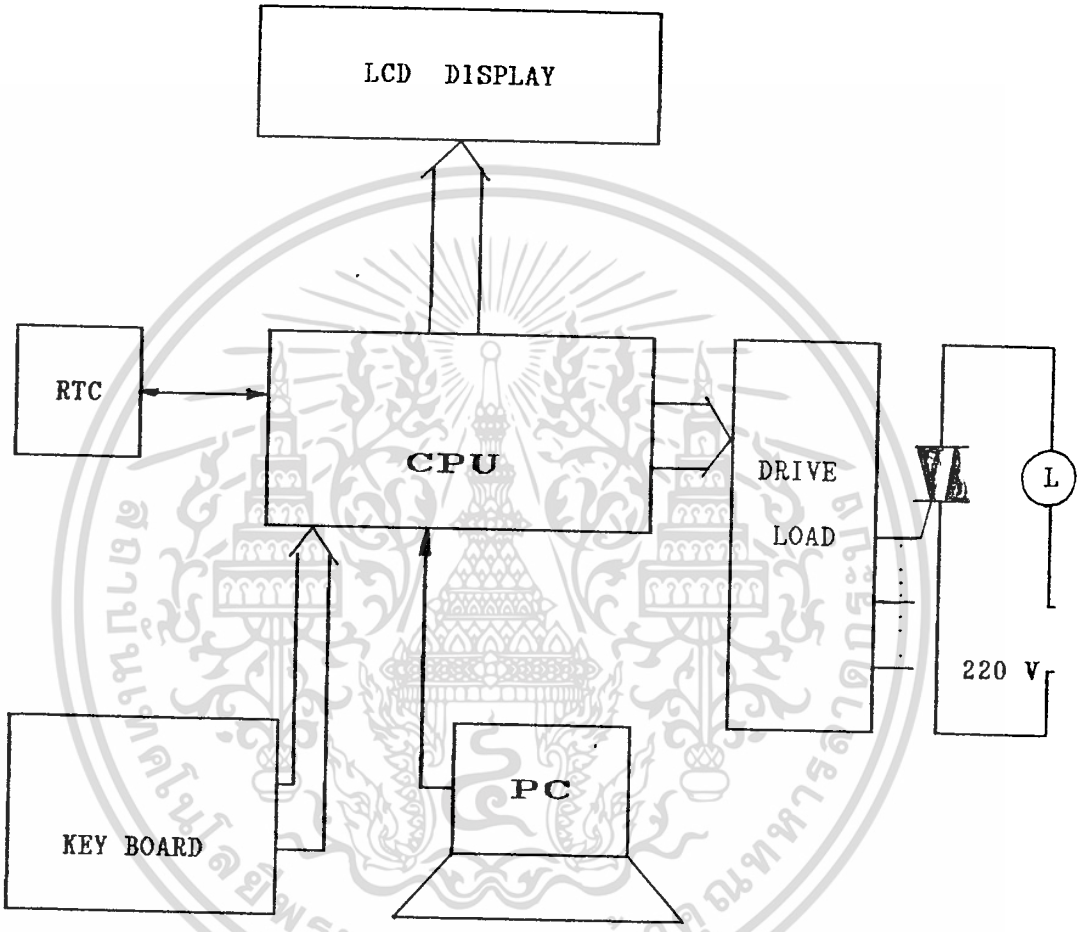
6. VSS เป็นขา GND

7. VO เป็นขารับแรงดันในการขับ LCD ให้ส่วางหรือมืด

บทที่ 3

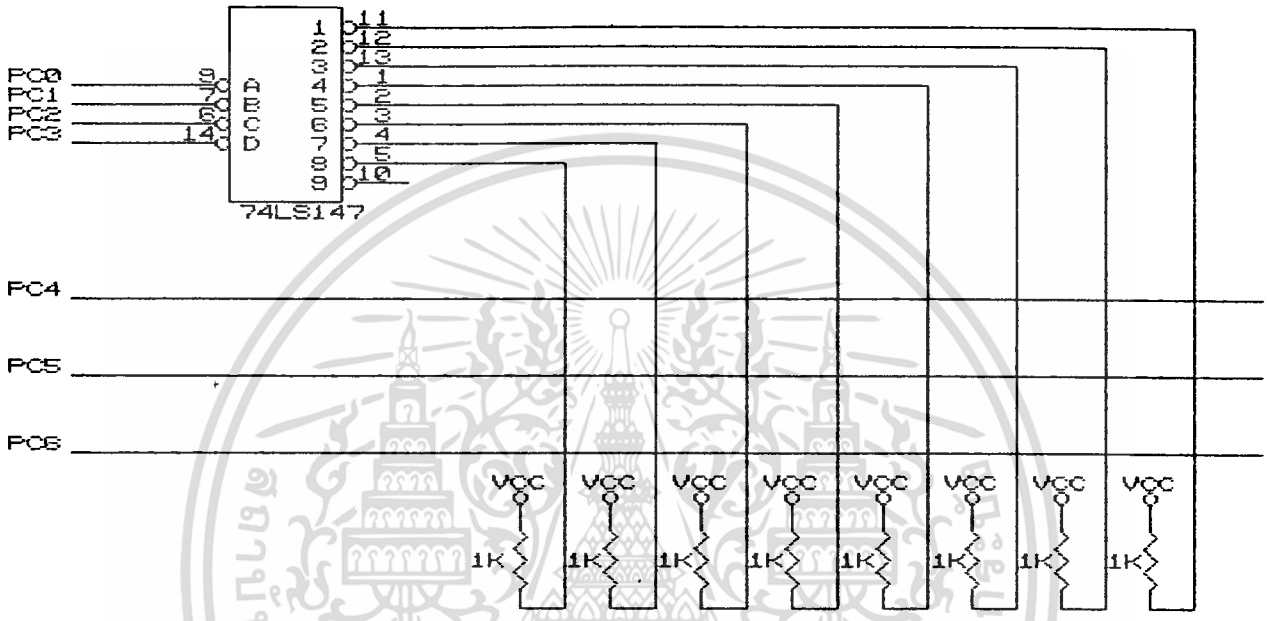
โครงสร้างเครื่องควบคุม

BLOCK DIAGRAM OF SYSTEM



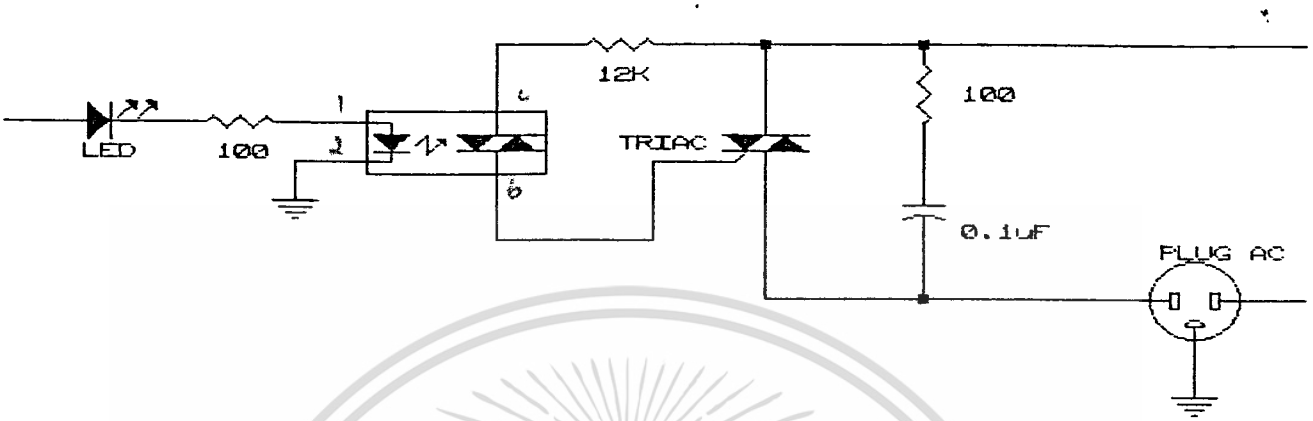
จาก BLOCKGRAM การทำงานของเครื่องควบคุมอุปกรณ์ไฟฟ้า โดยระบบทั้งหมดจะถูกควบคุมโดย CPU ทั้งหมด เราสามารถป้อนโปรแกรมการทำงานของโหลดช่องต่างๆผ่านทาง KEY BOARD หรือจะทำงานป้อนข้อมูลทางเครื่อง PC ก็ได้ ข้อดี ของการป้อนข้อมูลทาง PC คือสามารถ SAVE ข้อมูลลงแผ่น DISK ได้เมื่อต้องการก็ทำงาน LOAD เข้าไปใหม่ได้ นาฬิกาที่เครื่องใช้จะมาจากIC เบอร์ MM 58167 เป็น REAL TIME CLOCK (RTC) ข้อดีของIC เบอร์นี้คือการใช้งานสามารถทำได้ง่าย,มีความเที่ยงตรงสูง แม้แต่เวลาที่เราทำการปิดเครื่องนาฬิกาก็ยังสามารถเดินต่อไปอยู่เนื่องว่าในวงจรมีระบบ BACK UP ในส่วนของระบบควบคุมกระแสของ LOAD ใช้หลักการของ CURRENT TARNSFORMER นำมาต่อใช้งานร่วมกับระบบ ANALOG TO DIGITAL โดยทำการ INTERFACE เข้ากับ CPU นำค่าไปทำการคำนวณกระแสของโหลดต่อไป วงจรส่วนที่ใช้ในการขับโหลด จะใช้ TRIAC กระแสสูงทำให้ไม่มีการสปาร์คของหน้าคอนแท็ค

วงจร KEY BOARD



การทำงานของวงจร

การทำงานของวงจร KEY BOARD จะถูกควบคุมโดย CPU โดยใช้หลักการของการ SCAN KEY PC0-PC2 เป็น OUTPUT PORT , PC4-PC7 เป็น INPUT PORT , IC 74147 เป็น 10-4 ENCODER เริ่มแรก จะให้ PC0 = "0",PC1 = "1" ถ้ามีการกด KEY ใน ROW 1 เส้นของ CLOUM นั้นจะมีค่าเท่ากับ "0" ป้อนเป็น INPUT ให้แก่ 74147 ส่งให้ CPU ต่อไปขั้นตอนต่อไปก็จะให้ PC0 = "1" , PC1 = "0" , PC2 = "1" เพื่อทำการตรวจ ROW2 ว่ามีการกด KEY อะไรอยู่ จะทำงานอย่างนี้จนครบทุก ROW และทุก COLUM



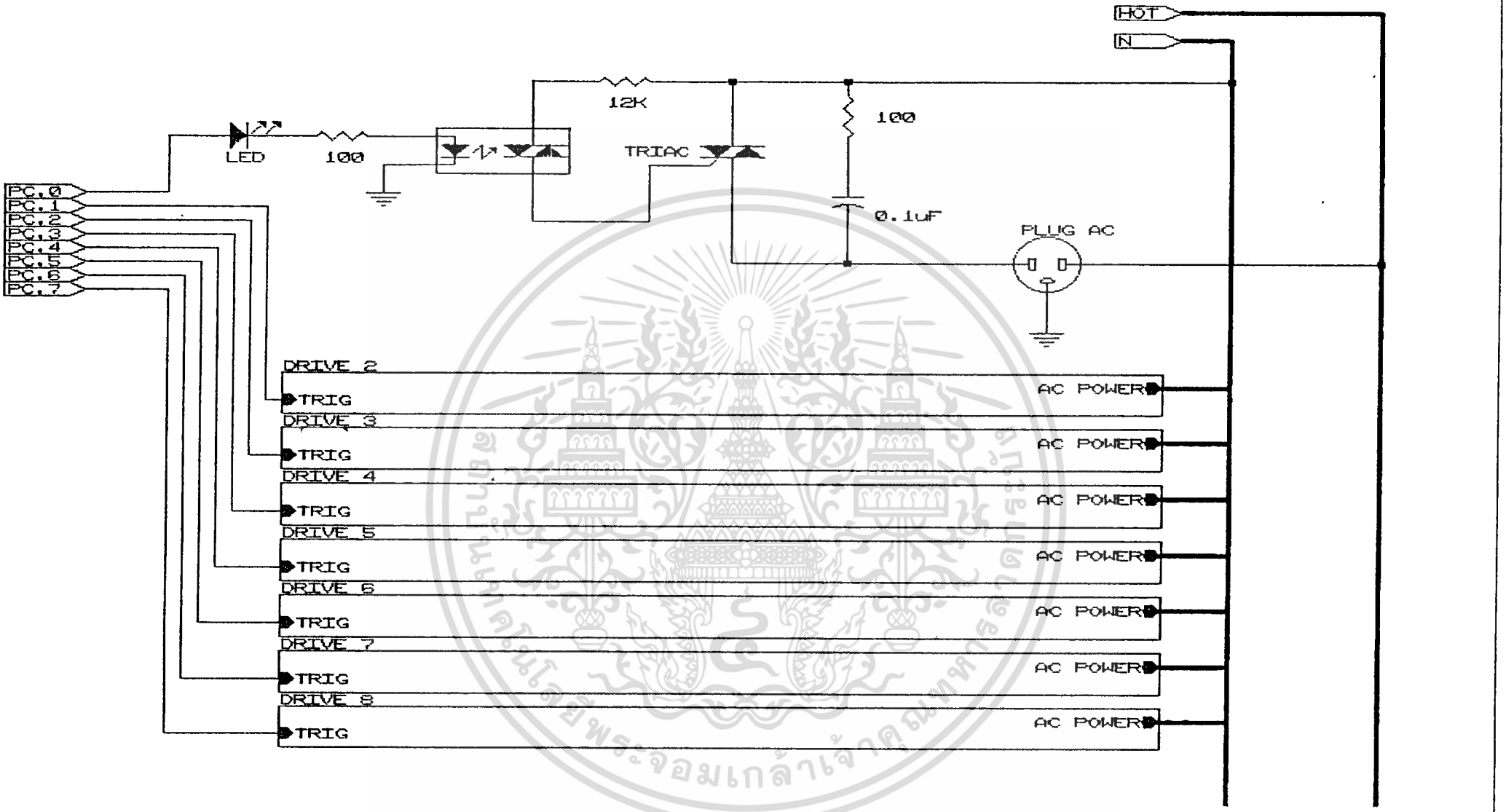
ระบบการขับโหลด (DRIVE LOAD)

ระบบการขับโหลดจะถูกควบคุมโดย CPU โดยมีเงื่อนไขตามโปรแกรมที่ป้อนเข้าไปก่อนแล้ว

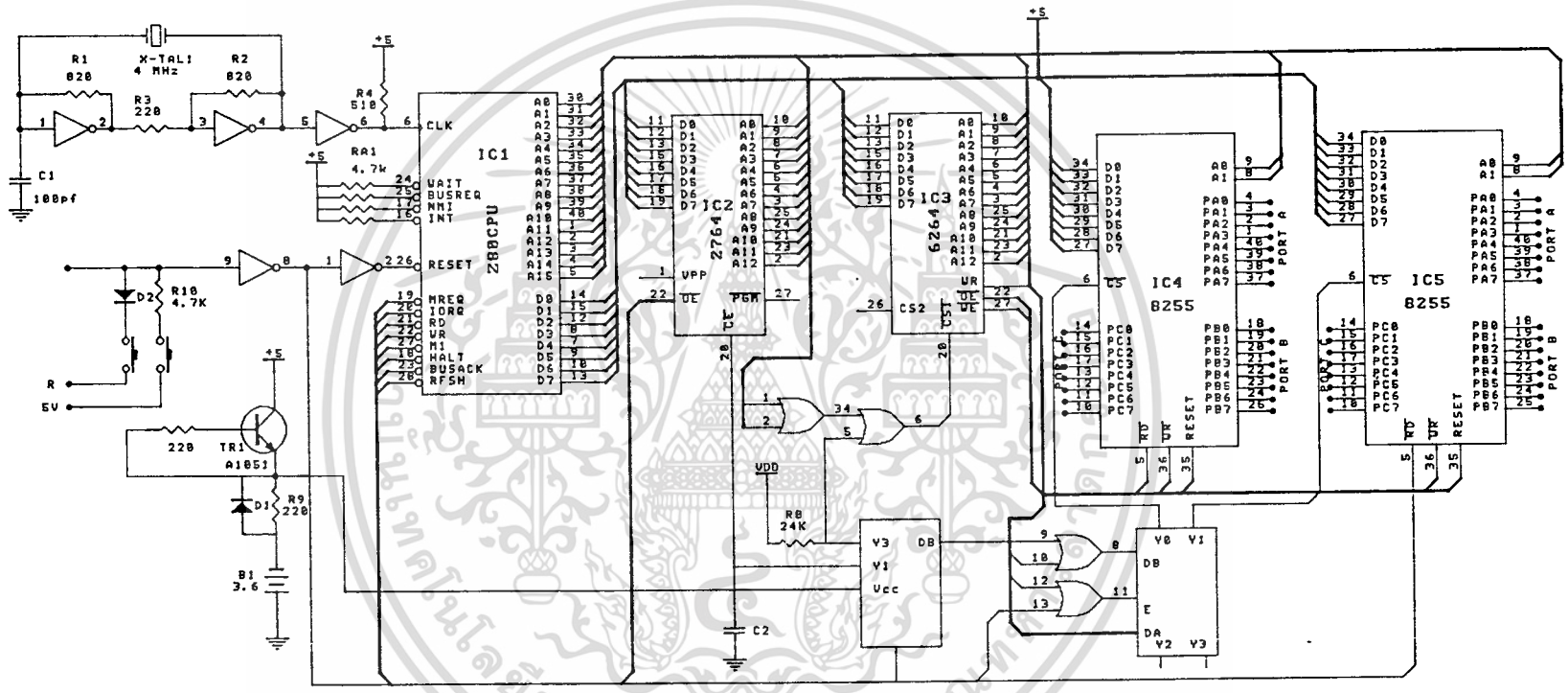
การทำงานของระบบ DRIVE LOAD

เมื่อ PC 0-7 ถ้ามีการส่งสัญญาณ "1" ผ่านขา 1 และ 2 ของ OP-TO DIAC ซึ่งจะทำให้มีสัญญาณออกทางขา 4-6 ของ OP-TO DIAC สัญญาณจะไปทริกขา G ของ TRIAC ซึ่งจะทำงาน สามารถไปขับโหลดที่ต่ออนุกรมกับ สาย LINE โดยผ่านตัว TRIAC ดังรูปวงจร DRIVE LOAD

ในชุด DRIVE LOAD จะมีการแบ่งออกเป็น 2 ประเภท ดังนี้ ประเภทแรกคือ PC 0-3 จะผ่านตัวหม้อแปลงที่ต่อ ขา MT2 ของ TRIAC ซึ่งมีหน้าที่วัดกระแสของโหลดให้ใช้หลักการของ CURRENT TRANSFORMER ประเภทที่สอง คือ PC 4-7 ทำหน้าที่ขับโหลดทั่วไปโดยไม่สามารถวัดกระแสโหลดได้ ชุด DRIVE LOAD ทุกชุดจะมีหลอด LED แสดงสภาวะการทำงานของ CHANNEL ต่างๆ คือ ถ้า LED สว่างแปลว่า CHANNEL นั้นกำลังทำงานอยู่ แต่ถ้า LED ดับแปลว่า CHANNEL นั้นไม่ได้ทำงาน

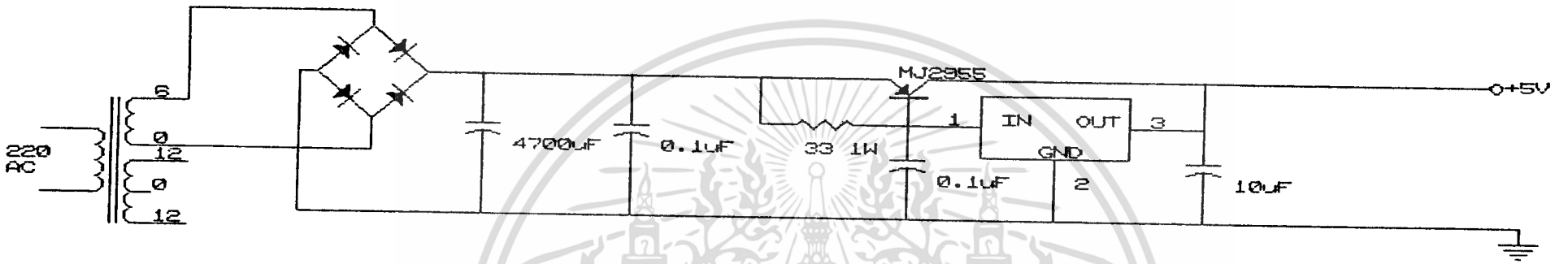


Size	Document Number	
A		
Date: January 1, 1980	Sheet	of



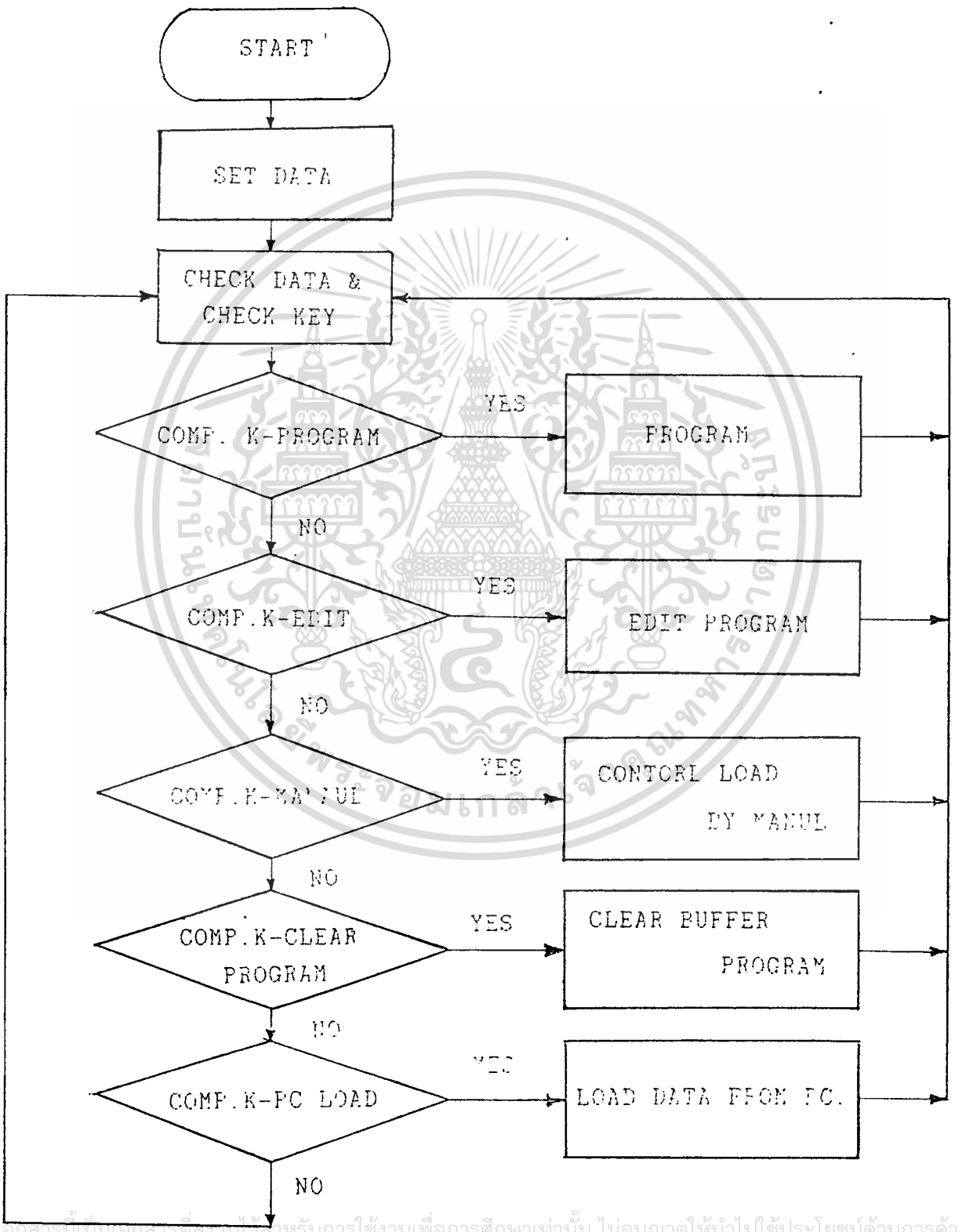
2-80 CONTROL BOARD

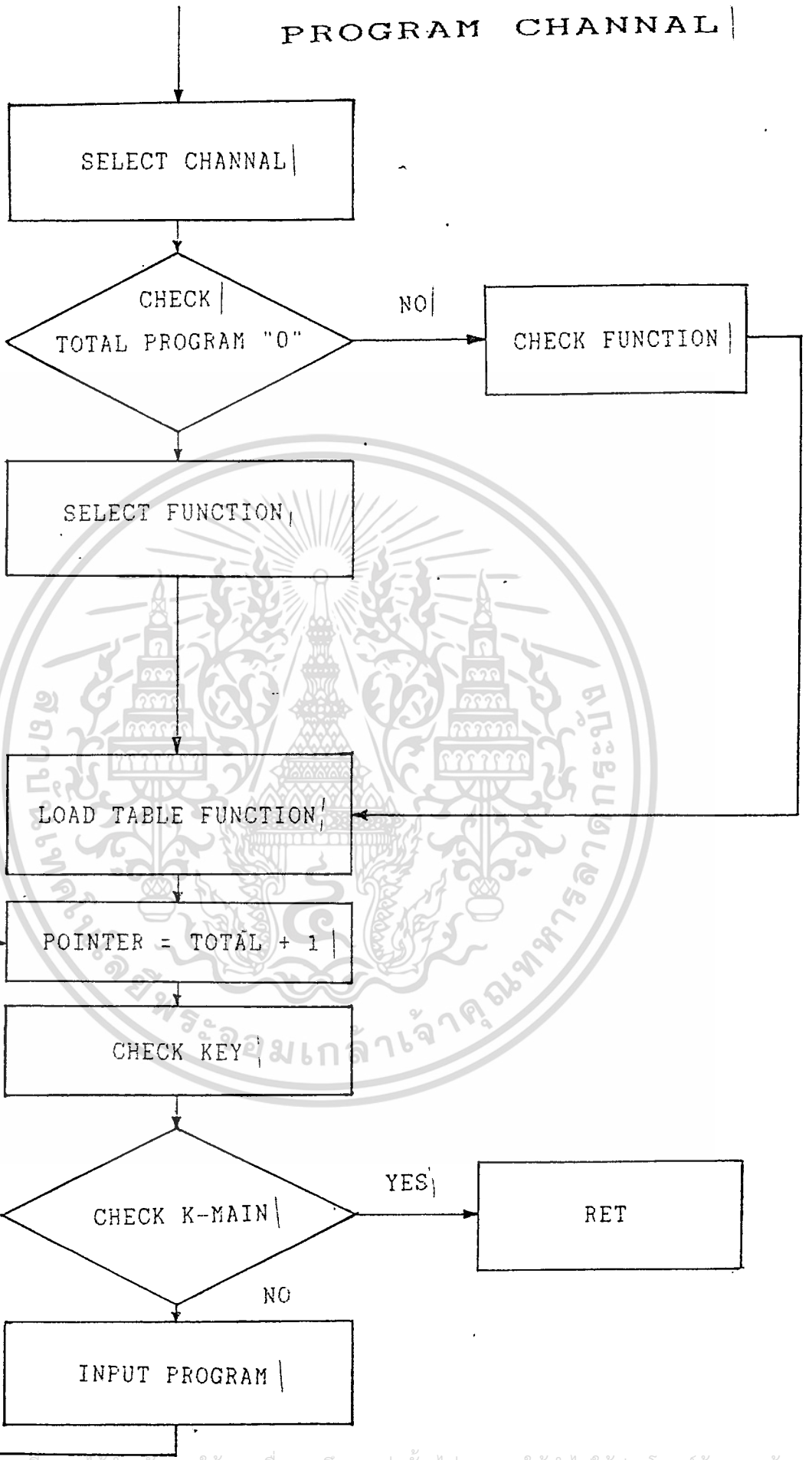
Title	2-80 CONTROL BOARD	
Size A3	Number	Revision
Date: 17-MAR 1992	Sheet of	
File: A:SERT3/1	Drawn By:	



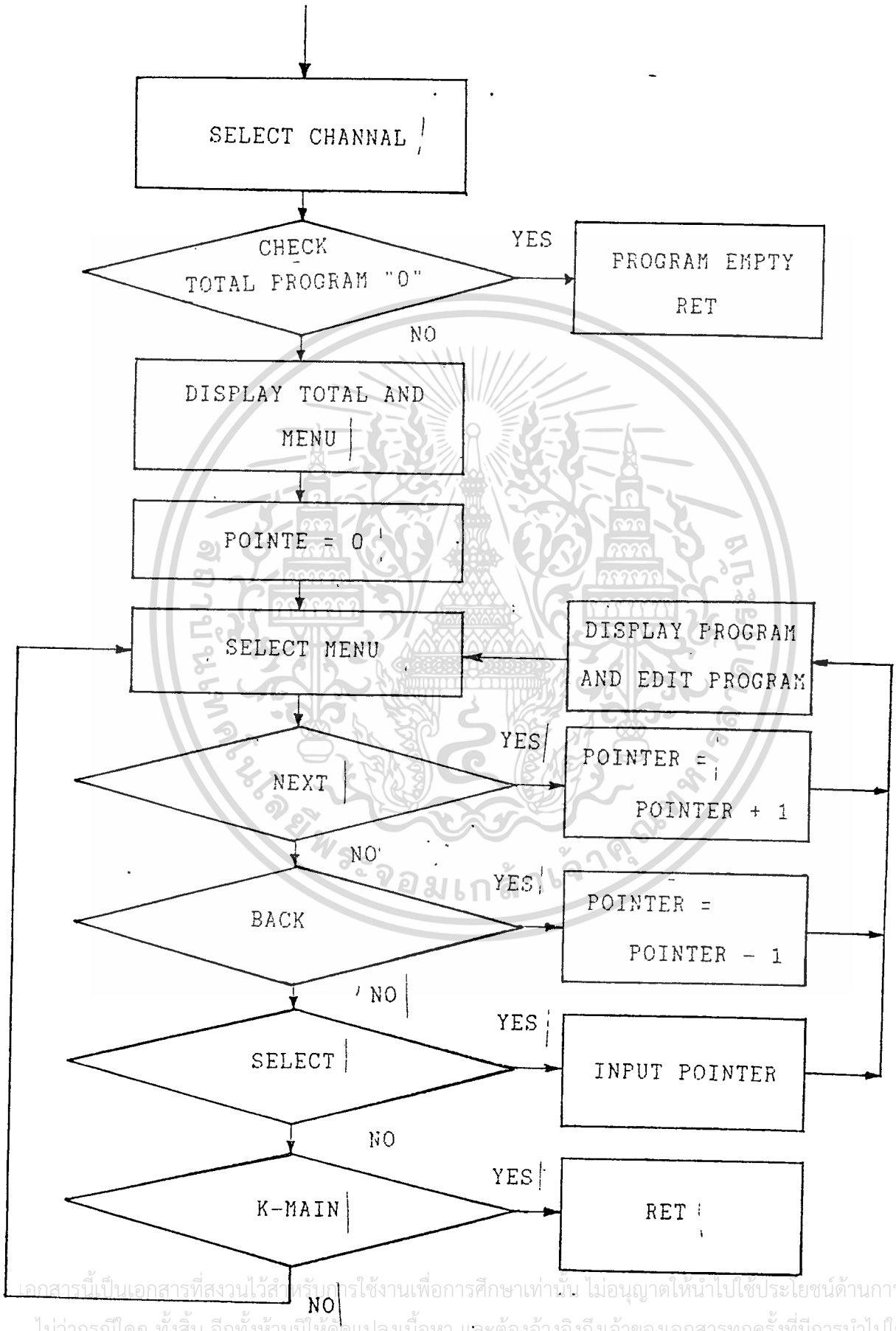
SUPPLY		
Size A	Document Number	F
Date: January 1, 1980	Sheet of	

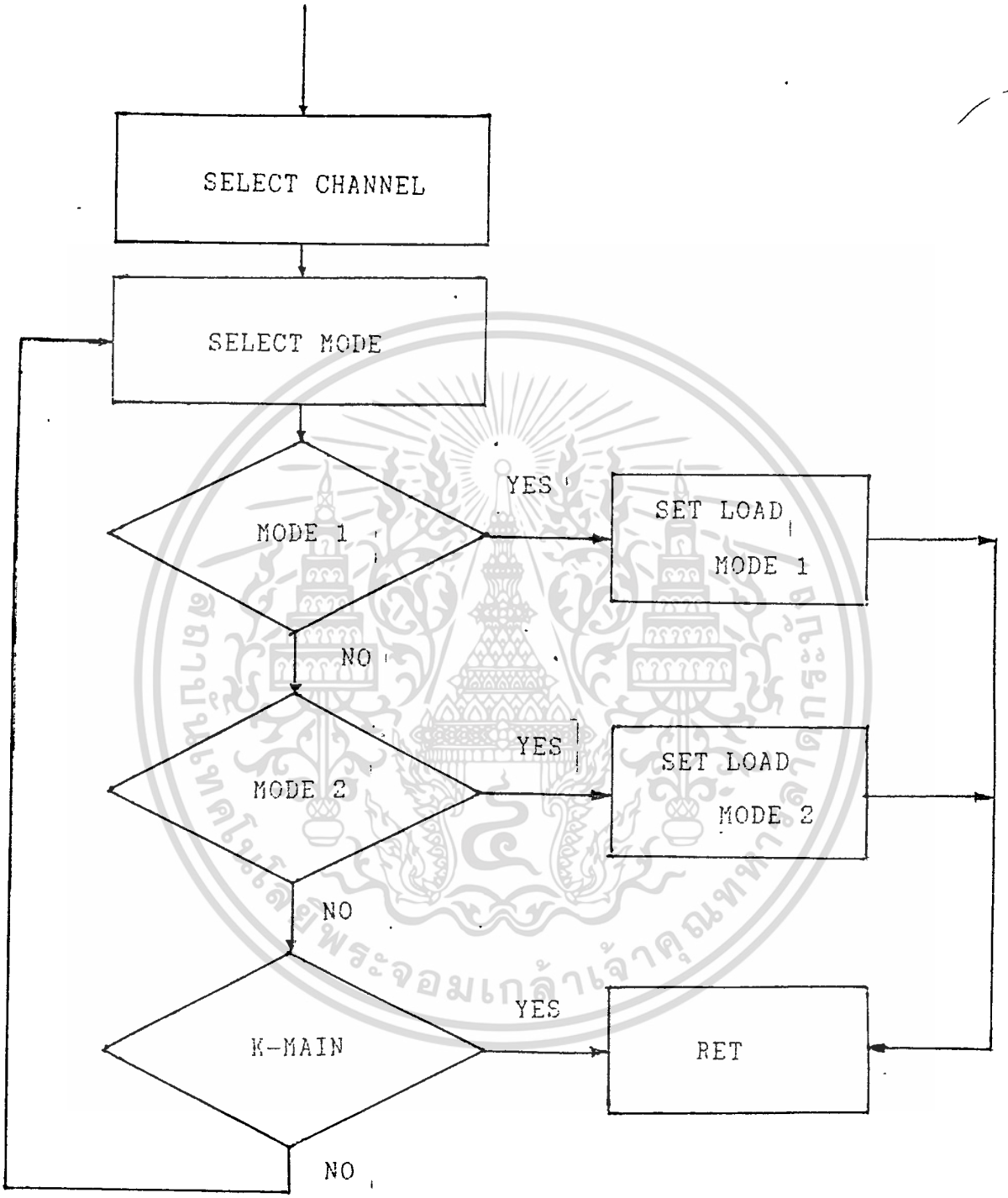
MAIN PROGRAM



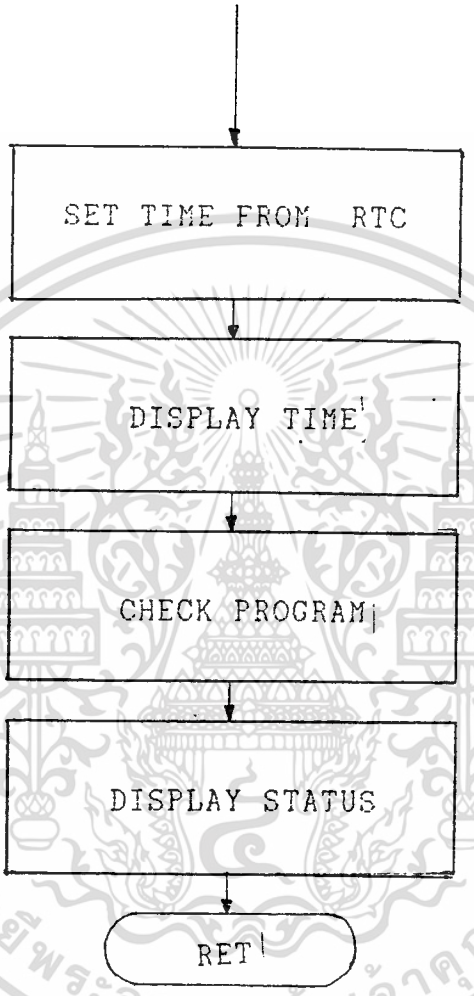


EDIT PROGRAM





INT MODE 1



การใช้งานเครื่อง

1.การเปิดเครื่อง

1.1 เปิดเครื่องโดยไม่มีการกดคีย์ใดๆ ข้อมูลที่ทำการโปรแกรมเอาไว้จะมีค่าเหมือนเดิม แต่โหลดจะออฟทุกตัว

1.2 เปิดเครื่องพร้อมกับกดคีย์ [DEC] เครื่องจะทำการลบข้อมูลทุกอย่างทั้งข้อมูลที่ทำการโปรแกรมเอาไว้,ค่ากระแสที่ตั้ง จะโดนลบหมด

2.การตั้งเวลาเครื่อง

ทำได้โดยการกดคีย์ [SET TIME] เครื่องจะเข้าสู่โหมดของการตั้งเวลา เครื่องจะแสดงเวลาปัจจุบันของเครื่องออกมาให้

Mon-Date-Day Hour:Min:Sec

Set new time

การตั้งเวลาของเครื่อง เช่น ต้องการตั้งวันที่ใหม่ก็ทำการกดคีย์ [Left] หรือ [Right] ให้ไปอยู่ที่ Date แล้วกดคีย์ [INC],[DEC] หรือ กดคีย์ตัวเลขเพื่อทำการตั้งเวลาเลยก็ได้

3.การโปรแกรมเวลาเครื่อง

ทำได้โดยการกดคีย์ [PROGRAM] มีขั้นตอนในการโปรแกรมดังนี้

3.1 เมื่อทำการกดคีย์ [PROGRAM] เครื่องจะทำการถามว่าจะโปรแกรมการทำงานของโหลดตัวไหน โดยจะมีเมนูมาให้ทำการเลือกดังนี้

Channel 12345678

การเลือกใช้คีย์ [Left] และ [Right] เมื่อได้ตามที่ต้องการแล้วกดคีย์ Enter เพื่อทำการเลือก

3.2 ถ้าเป็นการป้อนโปรแกรมของโหลดตัวนั้นเป็นครั้งแรก เครื่องก็จะทำการถามฟังก์ชั่นของการทำงานต่ออีกที แต่ถ้าโหลดถูกโปรแกรมมาแล้วเครื่องจะไม่ถามฟังก์ชั่นจะไปข้อ3.3 ต่อไปเลย การเลือกฟังก์ชั่นของการทำงานก็ทำได้เช่นเดียวกับข้อ 3.1

3.3 เครื่องจะแสดงจำนวนของโหลด ทางจอภาพให้ดูก่อนถ้าพร้อมทำการกดคีย์ Enter

3.4 ทำการโปรแกรมตามฟังก์ชั่นของโหลดนั้น

รูปแบบของฟังก์ชั่น

Function 1 Hour:Minute:Second

Function 2 Day Hour:Minute:Second

Function 3 Date Hour:Minute:Second

Function 4 Month date Hour:Minute:Second

การโปรแกรมเวลาของโหลดใช้ได้ทั้งคีย์ [Left],[Right],[INC],[DEC]หรือจะทำการกดคีย์ดังเลยไปเลยก็ได้ ถ้าทำการป้อนโปรแกรมผิดพลาดเครื่องจะทำการบอกสิ่งที่ผิดพลาดให้ทำการแก้ไข เมื่อทำการโปรแกรมเรียบร้อยกดคีย์ Enter เครื่องจะกลับไปทำงานในข้อ 3.3 จนกว่าจะกดคีย์ [MAIN] เครื่องก็จะกลับออกจากโหมดการโปรแกรมเวลา

4.การแก้ไขโปรแกรม

กดคีย์ [EDIT] เพื่อเข้าสู่โหมดการแก้ไขโปรแกรม

4.1 เมื่อทำการกดคีย์ [EDIT] เครื่องจะทำการถามว่าจะทำการแก้ไขโปรแกรมการทำงานของโหลดตัวไหน โดยจะมีเมนูมาให้ทำการเลือกดังนี้

Channel 12345678

การเลือกใช้คีย์ [Left] และ [Right] เมื่อได้ตามที่ต้องการแล้วกดคีย์ Enter เพื่อทำการเลือก

4.2 เครื่องจะบอกแสดงจำนวนโปรแกรมของโหลดตัวนั้นว่าทั้งหมดมีเท่าไหร่

4.3 เครื่องจะแสดงข้อมูลของโปรแกรมที่ 1 ก่อนจากนั้นก็ทำการกดคีย์ [INC] [DEC] จนกว่าจะได้โปรแกรมที่จะทำการแก้ไข

4.4 การแก้ไขก็กดคีย์ [PROGRAM] การโปรแกรมก็จะทำเหมือนโหมดการโปรแกรมทุกอย่าง

4.5 การลบเฉพาะโปรแกรมทำการกดคีย์ [CLEAR] เครื่องจะทำการลบเฉพาะโปรแกรมที่แสดงอยู่หน้าจอเท่านั้น

4.6 การออกจากโหมดการแก้ไขทำได้โดยการกดคีย์ [MAIN]

5.การเปิด-ปิด โหลดแบบโดยตรง

กดคีย์ [MANUAL] เครื่องจะเข้าสู่โหมด Manual

ทำการเลือกโหลด และจะมีเมนูให้ทำการเลือก

ON ทำการออนโหลดโดยตรงไม่สนใจว่าโปรแกรมจะจะตั้งอย่างไร

OFF ทำการออฟโหลดโดยตรงไม่สนใจว่าโปรแกรมจะจะตั้งอย่างไร

CANCEL ให้โหลดกลับไปทำงานตามโปรแกรมที่ทำการตั้งเอาไว้

6.การลบโปรแกรมเพื่อเริ่มทำการป้อนโปรแกรมใหม่

กดคีย์ [CLEAR] เครื่องจะเข้าสู่โหมดการลบโปรแกรม

ทำการเลือกโหลด และจะมีเมนูให้ทำการเลือกโหลดเมื่อต้องการลบก็กดคีย์ Enter ถ้าไม่ต้องการลบก็กดคีย์ [MAIN]

ข้อควรระวังในโหมดนี้คือ เครื่องจะทำการลบโปรแกรมทั้งหมดของโหลดตัวนั้นถ้าต้องการลบเฉพาะบางโปรแกรมของโหลดก็ใช้การลบโปรแกรมในโหมดการแก้ไขจะดีกว่า

7.การตั้งค่ากระแสที่ต้องการควบคุม

กดคีย์ [CURRENT] เพื่อเข้าสู่โหมดการตั้งกระแส

ทำการเลือกโหลด และจะมีเมนูให้ทำการเลือก

ON ทำการควบคุมกระแสและตั้งกระแส

OFF ไม่ควบคุมกระแสจะปล่อยโหลดให้เป็นอิสระ

8.การดูค่ากระแสของโหลดว่ามีค่าเท่าไหร่

กดคีย์ [E-ch] เพื่อขอดูค่าของกระแสที่ตั้งและค่าของกระแสที่วัดได้ว่ามีค่าเท่าไหร่

9.การป้อนข้อมูลจากเครื่องคอมพิวเตอร์ XT/AT Compactible

กดคีย์ [LINK] เพื่อเข้าสู่โหมดการรับข้อมูลจากเครื่อง PC แล้วกด Enter

1	2	3	PRO	ECH	ESC
4	5	6	EDIT	TME	PC
7	8	9	CLR	INC	DEC
MA	0	CUT	ENT	<—	—>

1-9 เป็นหมายเลขทั่วไป

MA. ย่อมาจาก MANUAL

CUT ย่อมาจาก CURRENT

ENT ย่อมาจาก ENTER

CLR ย่อมาจาก CLEAR

EDIT ย่อมาจาก EDIT PROGRAM

PRO ย่อมาจาก PROGRAM

ECH ย่อมาจาก EACH CHANNAL

TME ย่อมาจาก SET TIME

INC ย่อมาจาก INCREMENT

DEC ย่อมาจาก DECREMENT

ESC ย่อมาจาก MAIN PROGRAM

PC ย่อมาจาก LINK COMPUTER (PC.)

บทที่ 5

บทสรุปและวิจารณ์

จากการทำ project ชิ้นนี้ได้พบปัญหามากมายและปัญหาส่วนใหญ่เกิดในทาง hard ware เช่น ในการวัดกระแสของโหลดซึ่งเราคิดทำเพิ่มขึ้นเพื่อให้งานของได้สมบูรณ์แบบแต่ก็ปัญหาคือ ตัวหม้อแปลงที่ใช้ในการวัดกระแสโหลดเราพันขึ้นเองจึงเกิดปัญหาด้านเสถียรภาพในตัวหม้อแปลงต่ำเป็นผลทำให้เกิดการ loss ในตัวมากและยังเกิดการ induce ระหว่างหม้อแปลงกันเองอีก ในการแก้ที่เราคือต้องทำชุดวัดกระแสเพียงชุดเดียวเท่านั้นหรือไม่นั้นจะต้องใช้หม้อแปลงที่พันแบบ ฟาราเดย์ แล้วต่อลง ground แต่ใน project ผู้จัดทำใช้แก้โดยวิธีแรกซึ่งได้ผลดี ส่วนชุด drive ที่ใช้ triac ในการขับโหลดก็เกิดปัญหาว่าหาจุดทริกไม่ได้เนื่องจากว่า เกิด spice สูงมากหลายเท่าตัวจึงแก้โดยใช้ รีเลย์ แทนก็ได้ดี ปัญหาที่ส่วนใหญ่ทำให้เราเสียเวลาจึงไม่สามารถพัฒนา project ให้ได้อย่างที่หวังไว้ได้

จากปัญหาต่างๆเหล่านี้ผู้จัดทำว่า น่าจะเป็นประโยชน์สำหรับผู้ที่จะพัฒนา project นี้และน่าเป็นประโยชน์กับผู้ที่นำ project นี้ไปใช้งาน

ภาคผนวก

```
          CPU   "Z80.TBL"
          HOF   "BIN8"
          ORG   0000H
;*************************
;*  Port for control RTC *
;*************************
rtc:      equ   80h
sec1000:  equ   rtc
sec10:    equ   rtc+1
sec:      equ   rtc+2
minute:   equ   rtc+3
hour:     equ   rtc+4
day:      equ   rtc+5
date:     equ   rtc+6
month:    equ   rtc+7
r_sec100: equ   rtc+8
r_sec10:  equ   rtc+9
r_sec:    equ   rtc+0ah
r_min:    equ   rtc+0bh
r_hour:   equ   rtc+0ch
r_day:    equ   rtc+0dh
r_date:   equ   rtc+0eh
r_mon:    equ   rtc+0fh
t_int:    equ   rtc+10h
con_int:  equ   rtc+11h
t_reset:  equ   rtc+12h
r_reset:  equ   rtc+13h
sta_bit:  equ   rtc+14h
t_go:     equ   rtc+15h
stan_int: equ   rtc+16h
t_test:   equ   rtc+1fh
;      8255 (1)
P_adc:    equ   00h
P_b:      equ   01h
P_load:   equ   02h  ;drive LOAD
P_con1:   equ   03h
;      8255 (2)
P_lcd1:   equ   10h  ;for LCD
P_lcd2:   equ   11h  ;for LCD
P_key:    equ   12h  ;for Key board
P_con2:   equ   13h
;**************
;*  Data key  *
;**************
k_1:      equ   09bh ;1
k_2:      equ   0abh ;2
k_3:      equ   07bh ;3
k_pro:    equ   08bh ;program
k_5:      equ   0dbh ;4
k_6:      equ   0ebh ;5
k_7:      equ   0bbh ;6
k_edit:   equ   0cbh ;edit
k_9:      equ   09dh ;7
k_10:     equ   0adh ;8
k_11:     equ   07dh ;9
k_clear:  equ   08dh ;Clear
k_manual: equ   0ddh ;
k_14:     equ   0edh ;0
k_15:     equ   0bdh ;
k_enter:  equ   0cdh ;Enter
k_mem:    equ   07eh ;Memory
k_main:   equ   08eh ;Main
k_time:   equ   09eh ;Set time
k_pcl:    equ   0aeh ;load from PC
k_inc:    equ   0beh ;Inc
k_dec:    equ   0ceh ;dec
k_left:   equ   0deh ;left
k_right:  equ   0eeh ;right

;**********************
;*  Buffer ram data   *
;**********************
ram:      equ   8000h
b_sec10:  equ   ram
b_sec:    equ   ram+1
b_min:    equ   ram+2
b_hour:   equ   ram+3
b_day:    equ   ram+4
b_date:   equ   ram+5
b_mon:    equ   ram+6
comp:     equ   ram+7
key:      equ   ram+8  ;key board
disb_time: equ  ram+9  ;00:00:00
old_hour: equ   ram+18
old_min:  equ   ram+19
old_sec:  equ   ram+20
s_g1:     equ   ram+21  ;variable 1
s_g2:     equ   ram+22  ;variable 2
s_g3:     equ   ram+23  ;variable 3
s_g4:     equ   ram+24  ;variable 4
s_g5:     equ   ram+25  ;variable 5
s_g6:     equ   ram+26  ;variable 6
addr:     equ   ram+27  ;
addr2:    equ   ram+29  ;
add_k:    equ   ram+31
addr3:    equ   ram+34
b_set_time: equ ram+40
p_mon:    equ   ram+41
p_date:   equ   ram+42
p_day:    equ   ram+43
p_hour:   equ   ram+44
p_min:    equ   ram+45
p_sec:    equ   ram+46
bin_code: equ   ram+50
bcd_code: equ   ram+54
bcd_code1: equ  ram+58
b_status: equ   ram+65
b_manual1: equ  ram+66
b_manual2: equ  ram+67
dis_st:   equ   ram+68
mode:     equ   ram+80
buff_cur1: equ  ram+81
buff_pro1: equ  ram+100h
buff_pro2: equ  buff_pro1+0500h
buff_pro3: equ  buff_pro2+0500h
buff_pro4: equ  buff_pro3+0500h
buff_pro5: equ  buff_pro4+0500h
buff_pro6: equ  buff_pro5+0500h
buff_pro7: equ  buff_pro6+0500h
buff_pro8: equ  buff_pro7+0500h
```

```
begin:      jp    start
;**********************************
;*Program interupt data from RS-232*
;* OUT  = (s_g6) memory data       *
;*       = (s_g5) strobe           *
;**********************************
            ORG   0066h
            push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            in    a,(p_adc)
            ld    (s_g6),a
            ld    a,1
            ld    (s_g5),a
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            retn
start:      ld    sp,0fffeh
            call  wait
            ld    a,92h
            out   (p_con1),a
            ld    a,88h
            out   (P_con2),a
            call  initlcd
            call  off_cursor
            call  scan_key
            cp    k_right
            jp    z,begin_3
            cp    k_dec
            jp    z,begin_1
            call  power_on
            call  test_ram
            jp    begin_3
begin_1:    call  clrscr
            call  delay
            call  off_cursor
            ld    b,1
            ld    c,0
            ld    hl,s1_begin
            call  print
            call  wait
            call  wait
            call  wait
            call  wait
            call  power_on
            call  test_ram
            ld    a,0
            ld    (buff_pro1),a
            ld    (buff_pro2),a
            ld    (buff_pro3),a
            ld    (buff_pro4),a
            ld    (buff_pro5),a
            ld    (buff_pro6),a
            ld    (buff_pro7),a
            ld    (buff_pro8),a
            ld    (buff_pro1+2),a
            ld    (buff_pro2+2),a
            ld    (buff_pro3+2),a
            ld    (buff_pro4+2),a
            ld    (buff_pro5+2),a
            ld    (buff_pro6+2),a
            ld    (buff_pro7+2),a
            ld    (buff_pro8+2),a
            ld    a,0
            ld    (b_status),a
            ld    (b_manual1),a
            ld    (b_manual2),a
            out   (p_load),a
            ld    a,0ffh
            ld    (key),a
            xor   a
            out   (con_int),a
            ld    hl,0
            ld    (addr),hl
            jp    start1
begin_3:
            ld    a,0ffh
            ld    (key),a
            xor   a
            out   (con_int),a
            ld    a,0
            ld    (b_status),a
            ld    (b_manual1),a
            ld    (b_manual2),a
            out   (p_load),a
            ld    a,0ffh
            ld    (key),a
            ld    hl,0h
            ld    (addr),hl
start1:     call  time
            call  dis_date
            call  dis_time
            call  chk_pro
            call  status
            call  scan_key
            cp    k_time
            call  z,set_time
            cp    k_pro
            call  z,program
            cp    k_clear
            call  z,clear_pro
            cp    k_edit
            call  z,edit_pro
            cp    k_manual
            call  z,manual
            cp    k_pcl
            call  z,pcl
            jp    start1
pcl:        push  af
            push  bc
            push  de
            push  hl
            push  ix
```

```
        push  iy
        call  clrscr
        call  delay
        call  off_cursor
        ld    b,1
        ld    c,0
        ld    hl,s1_pcl
        call  print
        ld    b,2
        ld    c,0
        ld    hl,s2_pcl
        call  print
pcl_1:  call  scan_key
        cp    k_main
        jp    z,pcl_e
        cp    k_enter
        jp    z,pcl_2
        jp    pcl_1
pcl_2:  call  clrscr
        call  delay
        ld    b,1
        ld    c,0
        ld    hl,s3_pcl
        call  print
        ld    a,0
        ld    (s_g5),a
        ld    (s_g4),a
        ld    iy,buff_pro1
        ld    ix,buff_pro1
pcl_3:  call  scan_key
        cp    k_main
        jp    z,pcl_e
        ld    a,(s_g5)
        cp    1
        jp    z,pcl_5
        jp    pcl_3
pcl_4:  push  af
        push  bc
        push  de
        push  hl
        push  ix
        push  iy
        ld    a,40h
        call  gotoxy
        push  iy
        pop   hl
        ld    c,h
        call  hex_ascii
        call  wrbyte    ;0
        ld    d,e
        call  wrbyte    ;0
        ld    c,1
        call  hex_ascii
        call  wrbyte    ;0
        ld    d,e
        call  wrbyte    ;0
        ld    d,":"
        call  wrbyte    ;:
        ld    d," "
        call  wrbyte    ;space
        ld    a,(s_g6)
        ld    c,a
        call  hex_ascii
        call  wrbyte
        ld    d,e
        call  wrbyte
        pop   iy
        pop   ix
        pop   hl
        pop   de
        pop   bc
        pop   af
        ret
; Command code
;    F1  =  Start code
;    F2  =  Jump  channel
;    F3  =  End loading.
;  s_g6 =  Data
pcl_5:  push  af
        push  bc
        push  de
        push  hl
        ld    a,(s_g6)
        cp    0f1h
        jp    z,pcl_6
        cp    0f2h
        jp    z,pcl_7
        cp    0f3h
        jp    z,pcl_54
        ld    a,(s_g4)
        cp    0f1h
        jp    z,pcl_load
        call  clrscr
        call  delay
        ld    b,1
        ld    c,0
        ld    hl,s4_pcl
        call  print
        call  pcl_4
pcl_51: call  scan_key
        cp    k_main
        jp    z,pcl_53
        jp    pcl_51
pcl_53: pop   hl
        pop   de
        pop   bc
        pop   af
        jp    pcl_e
pcl_54: call  clrscr
        call  delay
        ld    b,1
        ld    c,0
        ld    hl,s6_pcl
        call  print
        call  wait
        call  wait
        call  wait
        jp    pcl_53
pcl_52: ld    a,0
        ld    (s_g5),a
```

```
            pop   hl
            pop   de
            pop   bc
            pop   af
            jp    pcl_3
pcl_6:      call  clrscr
            call  delay
            ld    b,1
            ld    c,0
            ld    hl,s5_pcl
            call  print
            ld    a,0f1h
            ld    (s_g4),a
            jp    pcl_52
pcl_load:   ld    a,(s_g6)
            ld    (iy+0),a
            inc   iy
            jp    pcl_52
pcl_7:      push  iy
            pop   hl
            ld    de,500h
            add   hl,de
            push  hl
            pop   iy
            jp    pcl_3
pcl_e:      call  clrscr
            call  delay
            call  off_cursor
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
manual:     push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  clrscr
            call  delay
            call  on_cursor
            call  select_ch
            ld    a,(s_g6)
            cp    1
            jp    z,manual_e
            ld    b,1
            ld    c,0
            ld    hl,sman_1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,sman_2
            call  print
            ld    a,67
            call  gotoxy
            ld    c,1
            call  on_cursor
manual_1:   call  manual_t
            call  scan_key
            cp    k_right
            call  z,man_right
            cp    k_left
            call  z,man_left
            cp    k_main
            jp    z,manual_e
            cp    k_enter
            jp    z,manual_2
            jp    manual_1
manual_2:   ld    a,c
            cp    1
            jp    z,manual_21
            cp    2
            jp    z,manual_22
            cp    3
            jp    z,manual_23
            jp    manual_e
manual_21:  ld    ix,b_manual1
            ld    iy,b_manual2
            ld    a,(s_g1)
            cp    1
            jp    z,manual_31
            cp    2
            jp    z,manual_32
            cp    3
            jp    z,manual_33
            cp    4
            jp    z,manual_34
            cp    5
            jp    z,manual_35
            cp    6
            jp    z,manual_36
            cp    7
            jp    z,manual_37
            cp    8
            jp    z,manual_38
manual_22:  ld    ix,b_manual1
            ld    iy,b_manual2
            ld    a,(s_g1)
            cp    1
            jp    z,manual_41
            cp    2
            jp    z,manual_42
            cp    3
            jp    z,manual_43
            cp    4
            jp    z,manual_44
            cp    5
            jp    z,manual_45
            cp    6
            jp    z,manual_46
            cp    7
            jp    z,manual_47
            cp    8
            jp    z,manual_48
manual_23:  ld    ix,b_manual1
            ld    iy,b_manual2
            ld    a,(s_g1)
```

```
                cp    1
                jp    z,manual_51
                cp    2
                jp    z,manual_52
                cp    3
                jp    z,manual_53
                cp    4
                jp    z,manual_54
                cp    5
                jp    z,manual_55
                cp    6
                jp    z,manual_56
                cp    7
                jp    z,manual_57
                cp    8
                jp    z,manual_58
manual_31:      sett  0,(ix+0)
                sett  0,(iy+0)
                jp    manual_e
manual_32:      sett  1,(ix+0)
                sett  1,(iy+0)
                jp    manual_e
manual_33:      sett  2,(ix+0)
                sett  2,(iy+0)
                jp    manual_e
manual_34:      sett  3,(ix+0)
                sett  3,(iy+0)
                jp    manual_e
manual_35:      sett  4,(ix+0)
                sett  4,(iy+0)
                jp    manual_e
manual_36:      sett  5,(ix+0)
                sett  5,(iy+0)
                jp    manual_e
manual_37:      sett  6,(ix+0)
                sett  6,(iy+0)
                jp    manual_e
manual_38:      sett  7,(ix+0)
                sett  7,(iy+0)
                jp    manual_e
manual_41:      res   0,(ix+0)
                sett  0,(iy+0)
                jp    manual_e
manual_42:      res   1,(ix+0)
                sett  1,(iy+0)
                jp    manual_e
manual_43:      res   2,(ix+0)
                sett  2,(iy+0)
                jp    manual_e
manual_44:      res   3,(ix+0)
                sett  3,(iy+0)
                jp    manual_e
manual_45:      res   4,(ix+0)
                sett  4,(iy+0)
                jp    manual_e
manual_46:      res   5,(ix+0)
                sett  5,(iy+0)
                jp    manual_e
manual_47:      res   6,(ix+0)
                sett  6,(iy+0)
                jp    manual_e
manual_48:      res   7,(ix+0)
                sett  7,(iy+0)
                jp    manual_e
manual_51:      res   0,(iy+0)
                jp    manual_e
manual_52:      res   1,(iy+0)
                jp    manual_e
manual_53:      res   2,(iy+0)
                jp    manual_e
manual_54:      res   3,(iy+0)
                jp    manual_e
manual_55:      res   4,(iy+0)
                jp    manual_e
manual_56:      res   5,(iy+0)
                jp    manual_e
manual_57:      res   6,(iy+0)
                jp    manual_e
manual_58:      res   7,(iy+0)
                jp    manual_e
man_left:       push  af
                dec   c
                ld    a,c
                cp    0
                call  z,man_left1
                pop   af
                ret
man_left1:      ld    c,4
                ret
man_right:      push  af
                inc   c
                ld    a,c
                cp    4
                call  z,man_right1
                pop   af
                ret
man_right1:     ld    c,1
                ret
manual_t:       push  af
                ld    a,c
                cp    1
                call  z,manual_t1
                cp    2
                call  z,manual_t2
                cp    3
                call  z,manual_t3
                pop   af
                ret
manual_t1:      push  af
                ld    a,(man_t)
                call  gotoxy
                pop   af
                ret
manual_t2:      push  af
                ld    a,(man_t+1)
                call  gotoxy
                pop   af
                ret
manual_t3:      push  af
                ld    a,(man_t+2)
```

```
                call  gotoxy
                pop   af
                ret
manual_e:       call  clrscr
                call  delay
                call  off_cursor
                pop   iy
                pop   ix
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
man_t:          dfb   67,71,76
no_update:      push  af
                push  bc
                push  de
                push  hl
                push  ix
                push  iy
                ld    a,0
                ld    (s_g6),a
                ld    b,(iy+0)
                ld    a,(b_mon)
                cp    b
                jp    nz,no_up_e
                ld    b,(iy+1)
                ld    a,(b_date)
                cp    b
                jp    nz,no_up_e
                ld    b,(iy+2)
                ld    a,(b_day)
                cp    b
                jp    nz,no_up_e
                ld    b,(iy+3)
                ld    a,(b_hour)
                cp    b
                jp    nz,no_up_e
                ld    b,(iy+4)
                ld    a,(b_min)
                cp    b
                jp    nz,no_up_e
                ld    b,(iy+5)
                ld    a,(b_sec)
                cp    b
                jp    nz,no_up_e
                ld    a,1
                ld    (s_g6),a
                call  clrscr
                call  delay
                ld    b,1
                ld    c,0
                ld    hl,s_no
                call  print
                ld    b,2
                ld    c,0
                ld    hl,s_no1
                call  print
                call  wait
                call  wait
                call  wait
no_up_e:        pop   iy
                pop   ix
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
update:         push  af
                push  bc
                push  de
                push  hl
                push  ix
                push  iy
                ld    a,0
                ld    (s_g6),a
                ld    a,(b_mon)
                cp    13h
                jp    nc,update_1
                ld    a,(b_date)
                cp    32h
                jp    nc,update_2
                ld    a,(b_day)
                cp    8
                jp    nc,update_3
                ld    a,(b_hour)
                cp    24h
                jp    nc,update_4
                ld    a,(b_min)
                cp    60h
                jp    nc,update_5
                ld    a,(b_sec)
                cp    60h
                jp    nc,update_6
update_e:       pop   iy
                pop   ix
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
update_1:       ld    b,2
                ld    c,0
                ld    hl,sup_1
                call  print
                ld    a,(b_mon)
                jp    update_7
update_2:       ld    b,2
                ld    c,0
                ld    hl,sup_2
                call  print
                ld    a,(b_date)
                jp    update_7
update_3:       ld    b,2
                ld    c,0
                ld    hl,sup_3
                call  print
                ld    a,(b_day)
```

```
                jp    update_7
update_4:       ld    b,2
                ld    c,0
                ld    hl,sup_4
                call  print
                ld    a,(b_hour)
                jp    update_7
update_5:       ld    b,2
                ld    c,0
                ld    hl,sup_5
                call  print
                ld    a,(b_min)
                jp    update_7
update_6:       ld    b,2
                ld    c,0
                ld    hl,sup_6
                call  print
                ld    a,(b_sec)
                jp    update_7
update_7:       ld    c,a
                call  hex_ascii
                call  wrbyte
                ld    d,e
                call  wrbyte
                ld    a,1
                ld    (s_g6),a
                ld    d," "
                call  wrbyte
                call  wrbyte
                call  wait
                call  wait
                call  wait
                call  wait
                call  wait
                ld    b,2
                ld    c,0
                ld    hl,sup_7
                call  print
                ld    b,2
                ld    c,0
                ld    hl,st_pfll
                call  print
                jp    update_e
edit_pro:       push  af
                push  bc
                push  de
                push  hl
                push  ix
                push  iy
                call  clrscr
                call  delay
                ld    b,2
                ld    c,0
                ld    hl,se_1
                call  print
                call  select_ch
                ld    a,(s_g6)
                cp    1
                jp    z,edit_e
                ld    a,(ix+0)
                cp    0
                jp    z,edit_em
                call  clrscr
                call  delay
                call  off_cursor
                ld    b,1
                ld    c,0
                ld    hl,se_2
                call  print
                ld    c,(ix+1)
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    b,2
                ld    c,0
                ld    hl,se_3
                call  print
                ld    a,72
                call  gotoxy
                ld    b,(ix+0)
                call  bin_bcd
                ld    c,b
                call  hex_ascii
                call  wrbyte
                ld    d,e
                call  wrbyte
                call  wait
                call  wait
                call  wait
                call  wait
                call  wait
                call  wait
                call  clrscr
                call  delay
                ld    a,0
                ld    (s_g1),a
                ld    a,(ix+1)
                ld    (s_g2),a
edit_1:         ld    a,(s_g2)
                cp    1
                call  z,edit_3
                cp    2
                call  z,edit_4
                cp    3
                call  z,edit_5
                cp    4
                call  z,edit_6
edit_2:         call  scan_key
                cp    k_main
                jp    z,edit_e
                cp    k_inc
                jp    z,edit_inc
                cp    k_dec
                jp    z,edit_dec
                cp    k_pro
                jp    z,edit_pr
                cp    k_clear
                jp    z,edit_cl
                jp    edit_2
```

```
edit_cl:    ld    a,(s_g1)
            ld    b,a
            ld    a,0
            ld    a,(s_g6)
            ld    a,(ix+0)
            sub   b
            cp    1
            jp    z,edit_cl2
            cp    2
            jp    z,edit_cl3
            call  wait
            ld    b,a
            push  iy
edit_cl1:   ld    a,(iy+0ch)
            ld    (iy+0),a
            ld    a,(iy+0dh)
            ld    (iy+1),a
            ld    a,(iy+0eh)
            ld    (iy+2),a
            ld    a,(iy+0fh)
            ld    (iy+3),a
            ld    a,(iy+10h)
            ld    (iy+4),a
            ld    a,(iy+11h)
            ld    (iy+5),a
            ld    a,(iy+12h)
            ld    (iy+6),a
            ld    a,(iy+13h)
            ld    (iy+7),a
            ld    a,(iy+14h)
            ld    (iy+8),a
            ld    a,(iy+15h)
            ld    (iy+9),a
            ld    a,(iy+16h)
            ld    (iy+0ah),a
            ld    a,(iy+17h)
            ld    (iy+0bh),a
            call  inc_ch
            djnz  edit_cl1
            dec   (ix+0)
            pop   iy
            call  wait
            ld    a,(s_g6)
            cp    1
            call  z,dec_ch
            ld    a,(ix+0)
            cp    0
            jp    z,edit_em
            jp    edit_1
edit_cl2:   push  iy
            ld    a,(s_g1)
            dec   a
            ld    (s_g1),a
            ld    b,1
            ld    a,1
            ld    (s_g6),a
            jp    edit_cl1
edit_cl3:   push  iy
            ld    b,1
            jp    edit_cl1

edit_pr:    ld    a,(ix+1)
            cp    1
            jp    z,edit_p1
            cp    2
            jp    z,edit_p2
            cp    3
            jp    z,edit_p3
            cp    4
            jp    z,edit_p4
edit_p1:    push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  cl_baa
            call  clrscr
            call  delay
            call  on_cursor
            call  wait
            ld    b,1
            ld    c,0
            ld    hl,st_pf1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_pf11
            call  print
            ld    a,11
            call  gotoxy
            call  dis_f1
            ld    a,11
            call  gotoxy
            ld    c,1
edit_p11:   call  ta_p1
            call  scan_key
            cp    k_left
            call  z,p1_left
            cp    k_right
            call  z,p1_right
            cp    k_inc
            call  z,p1_inc
            cp    k_dec
            call  z,p1_dec
            cp    k_main
            jp    z,edit_p_e  ;new
            cp    k_enter
            jp    z,edit_p12  ;new
            call  key_press
            jr    edit_p11
edit_p12:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,edit_p1
            ld    a,(b_mon)
            ld    (iy+0),a
            ld    a,(b_date)
            ld    (iy+1),a
            ld    a,(b_day)
            ld    (iy+2),a
```

```
            ld    a,(b_hour)
            ld    (iy+3),a
            ld    a,(b_min)
            ld    (iy+4),a
            ld    a,(b_sec)
            ld    (iy+5),a
up_1:       ld    a,75
            call  gotoxy
            call  dis_f1
            ld    a,11
            call  gotoxy
            ld    c,1
edit_p13:   call  ota_p1
            call  scan_key
            cp    k_left
            call  z,p1_left
            cp    k_right
            call  z,p1_right
            cp    k_inc
            call  z,op1_inc
            cp    k_dec
            call  z,op1_dec
            cp    k_main
            jp    z,edit_p_e
            cp    k_enter
            jp    z,edit_p14
            call  key1_press
            jr    edit_p13

edit_p14:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,up_1
            ld    a,(b_mon)
            ld    (iy+6),a
            ld    a,(b_date)
            ld    (iy+7),a
            ld    a,(b_day)
            ld    (iy+8),a
            ld    a,(b_hour)
            ld    (iy+9),a
            ld    a,(b_min)
            ld    (iy+10),a
            ld    a,(b_sec)
            ld    (iy+11),a
edit_p_e:   call  wait
            call  clrscr
            call  delay
            call  off_cursor
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            jp    edit_1
edit_p2:    push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  cl_baa
            call  clrscr   ;function 1
            call  delay
            call  on_cursor
            call  wait
            ld    b,1
            ld    c,0
            ld    hl,st_pf1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_pf11
            call  print
            ld    a,7              ;
            call  gotoxy
            call  dis_f2           ;
            ld    a,7              ;
            call  gotoxy
            ld    c,1
edit_p21:   call  ta_p2            ;
            call  scan_key
            cp    k_left
            call  z,p2_left        ;
            cp    k_right
            call  z,p2_right       ;
            cp    k_inc
            call  z,p2_inc         ;
            cp    k_dec
            call  z,p2_dec         ;
            cp    k_main
            jp    z,edi2_p_e       ;new
            cp    k_enter
            jp    z,edit_p22       ;new
            call  key_press2
            jr    edit_p21         ;

edit_p22:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,edit_p2
            ld    a,(b_mon)
            ld    (iy+0),a
            ld    a,(b_date)
            ld    (iy+1),a
            ld    a,(b_day)
            ld    (iy+2),a
            ld    a,(b_hour)
            ld    (iy+3),a
            ld    a,(b_min)
            ld    (iy+4),a
            ld    a,(b_sec)
            ld    (iy+5),a
up_2:       ld    a,71             ;
            call  gotoxy
            call  dis_f2
            ld    a,7              ;
            call  gotoxy
            ld    c,1
edit_p23:   call  ota_p2           ;
```

```
            call  scan_key
            cp    k_left
            call  z,p2_left       ;
            cp    k_right
            call  z,p2_right      ;
            cp    k_inc
            call  z,op2_inc       ;
            cp    k_dec
            call  z,op2_dec       ;
            cp    k_main
            jp    z,edi2_p_e      ;
            cp    k_enter
            jp    z,edit_p24      ;
            call  key1_p2
            jr    edit_p23        ;

edit_p24:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,up_2
            ld    a,(b_mon)
            ld    (iy+6),a
            ld    a,(b_date)
            ld    (iy+7),a
            ld    a,(b_day)
            ld    (iy+8),a
            ld    a,(b_hour)
            ld    (iy+9),a
            ld    a,(b_min)
            ld    (iy+10),a
            ld    a,(b_sec)
            ld    (iy+11),a
edi2_p_e:   call  wait
            call  clrscr
            call  delay
            call  off_cursor
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            jp    edit_1
edit_p3:    push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  cl_baa
            call  clrscr    ;function 1
            call  delay
            call  on_cursor
            call  wait
            ld    b,1
            ld    c,0
            ld    hl,st_pf1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_pf11
            call  print
            ld    a,8               ;
            call  gotoxy
            call  dis_f3            ;
            ld    a,8               ;
            call  gotoxy
            ld    c,1
edit_p31:   call  ta_p3             ;
            call  scan_key
            cp    k_left
            call  z,p3_left         ;
            cp    k_right
            call  z,p3_right        ;
            cp    k_inc
            call  z,p3_inc          ;
            cp    k_dec
            call  z,p2_dec          ;
            cp    k_main
            jp    z,edi3_p_e        ;new
            cp    k_enter
            jp    z,edit_p32        ;new
            call  key_press3
            jr    edit_p31          ;
edit_p32:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,edit_p3
            ld    a,(b_mon)
            ld    (iy+0),a
            ld    a,(b_date)
            ld    (iy+1),a
            ld    a,(b_day)
            ld    (iy+2),a
            ld    a,(b_hour)
            ld    (iy+3),a
            ld    a,(b_min)
            ld    (iy+4),a
            ld    a,(b_sec)
            ld    (iy+5),a
up_3:       ld    a,72              ;
            call  gotoxy
            call  dis_f3
            ld    a,8               ;
            call  gotoxy
            ld    c,1
edit_p33:   call  ota_p3            ;
            call  scan_key
            cp    k_left
            call  z,p3_left         ;
            cp    k_right
            call  z,p3_right        ;
            cp    k_inc
            call  z,op3_inc         ;
            cp    k_dec
            call  z,op3_dec         ;
            cp    k_main
            jp    z,edi3_p_e        ;
            cp    k_enter
            jp    z,edit_p34        ;
```

```
            call  key3_p3
            jr    edit_p33        ;
edit_p34:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,up_3
            ld    a,(b_mon)
            ld    (iy+6),a
            ld    a,(b_date)
            ld    (iy+7),a
            ld    a,(b_day)
            ld    (iy+8),a
            ld    a,(b_hour)
            ld    (iy+9),a
            ld    a,(b_min)
            ld    (iy+10),a
            ld    a,(b_sec)
            ld    (iy+11),a
edi3_p_e:   call  wait
            call  clrscr
            call  delay
            call  off_cursor
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            jp    edit_1
edit_p4:    push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  cl_baa
            call  clrscr  ;function 1
            call  delay
            call  on_cursor
            call  wait
            ld    b,1
            ld    c,0
            ld    hl,st_pf1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_pf11
            call  print
            ld    a,5             ;
            call  gotoxy
            call  dis_f4          ;
            ld    a,69            ;
            call  gotoxy
            ld    c,1
edit_p41:   call  ta_p4           ;
            call  scan_key
            cp    k_left
            call  z,p4_left       ;
            cp    k_right
            call  z,p4_right      ;
            cp    k_inc
            call  z,p4_inc        ;
            cp    k_dec
            call  z,p4_dec        ;
            cp    k_main
            jp    z,edi4_p_e      ;new
            cp    k_enter
            jp    z,edit_p42      ;new
            call  key_press4
            jr    edit_p41        ;
edit_p42:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,edit_p4
            ld    a,(b_mon)
            ld    (iy+0),a
            ld    a,(b_date)
            ld    (iy+1),a
            ld    a,(b_day)
            ld    (iy+2),a
            ld    a,(b_hour)
            ld    (iy+3),a
            ld    a,(b_min)
            ld    (iy+4),a
            ld    a,(b_sec)
            ld    (iy+5),a
up_4:       ld    a,69            ;
            call  gotoxy
            call  dis_f4
            ld    a,5             ;
            call  gotoxy
            ld    c,1
edit_p43:   call  ota_p4          ;
            call  scan_key
            cp    k_left
            call  z,p4_left       ;
            cp    k_right
            call  z,p4_right      ;
            cp    k_inc
            call  z,op4_inc       ;
            cp    k_dec
            call  z,op4_dec       ;
            cp    k_main
            jp    z,edi4_p_e      ;
            cp    k_enter
            jp    z,edit_p44      ;
            call  key4_p4
            jr    edit_p43        ;
edit_p44:   call  update
            ld    a,(s_g6)
            cp    1
            jp    z,up_4
            ld    a,(b_mon)
            ld    (iy+6),a
            ld    a,(b_date)
            ld    (iy+7),a
            ld    a,(b_day)
            ld    (iy+8),a
            ld    a,(b_hour)
            ld    (iy+9),a
```

```
                ld    a,(b_min)
                ld    (iy+10),a
                ld    a,(b_sec)
                ld    (iy+11),a
edi4_p_e:       call  wait
                call  clrscr
                call  delay
                call  off_cursor
                pop   iy
                pop   ix
                pop   hl
                pop   de
                pop   bc
                pop   af
                jp    edit_1
cl_baa:         push  af
                ld    a,(iy+0)
                ld    (b_mon),a
                ld    a,(iy+1)
                ld    (b_date),a
                ld    a,(iy+2)
                ld    (b_day),a
                ld    a,(iy+3)
                ld    (b_hour),a
                ld    a,(iy+4)
                ld    (b_min),a
                ld    a,(iy+5)
                ld    (b_sec),a
                pop   af
                ret
edit_inc:       ld    a,(ix+0)
                ld    b,a
                ld    a,(s_gl)
                inc   a
                cp    b
                jp    z,edit_1
                call  inc_ch
                ld    a,(s_gl)
                inc   a
                ld    (s_gl),a
                jp    edit_1
edit_dec:       ld    a,0
                ld    b,a
                ld    a,(s_gl)
                cp    b
                jp    z,edit_1
                call  dec_ch
                ld    a,(s_gl)
                dec   a
                ld    (s_gl),a
                jp    edit_1
edit_e:         call  clrscr
                call  delay
                call  off_cursor
                pop   iy
                pop   ix
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
edit_3:         push  af
                call  clrscr
                call  delay
                ld    a,(s_gl)
                ld    c,a
                inc   c
                ld    b,c
                call  bin_bcd
                ld    c,b
                call  hex_ascii
                call  wrbyte
                ld    d,e
                call  wrbyte
                ld    a,11
                call  gotoxy
                ld    a,(iy+0)
                ld    (b_mon),a
                ld    a,(iy+1)
                ld    (b_date),a
                ld    a,(iy+2)
                ld    (b_day),a
                ld    a,(iy+3)
                ld    (b_hour),a
                ld    a,(iy+4)
                ld    (b_min),a
                ld    a,(iy+5)
                ld    (b_sec),a
                call  dis_fl
                ld    a,75
                call  gotoxy
                ld    a,(iy+6)
                ld    (b_mon),a
                ld    a,(iy+7)
                ld    (b_date),a
                ld    a,(iy+8)
                ld    (b_day),a
                ld    a,(iy+9)
                ld    (b_hour),a
                ld    a,(iy+0ah)
                ld    (b_min),a
                ld    a,(iy+0bh)
                ld    (b_sec),a
                call  dis_fl
                pop   af
                ret
edit_4:         push  af
                call  clrscr
                call  delay
                ld    a,(s_gl)
                ld    c,a
                inc   c
                ld    b,c
                call  bin_bcd
                ld    c,b
                call  hex_ascii
                call  wrbyte
                ld    d,e
                call  wrbyte
                ld    a,7
```

```
        call  gotoxy
        ld    a,(iy+0)
        ld    (b_mon),a
        ld    a,(iy+1)
        ld    (b_date),a
        ld    a,(iy+2)
        ld    (b_day),a
        ld    a,(iy+3)
        ld    (b_hour),a
        ld    a,(iy+4)
        ld    (b_min),a
        ld    a,(iy+5)
        ld    (b_sec),a
        call  dis_f2
        ld    a,71
        call  gotoxy
        ld    a,(iy+6)
        ld    (b_mon),a
        ld    a,(iy+7)
        ld    (b_date),a
        ld    a,(iy+8)
        ld    (b_day),a
        ld    a,(iy+9)
        ld    (b_hour),a
        ld    a,(iy+0ah)
        ld    (b_min),a
        ld    a,(iy+0bh)
        ld    (b_sec),a
        call  dis_f2
        pop   af
        ret
edit_5: push  af
        call  clrscr
        call  delay
        ld    a,(s_gl)
        ld    c,a
        inc   c
        ld    b,c
        call  bin_bcd
        ld    c,b
        call  hex_ascii
        call  wrbyte
        ld    d,e
        call  wrbyte
        ld    a,8
        call  gotoxy
        ld    a,(iy+0)
        ld    (b_mon),a
        ld    a,(iy+1)
        ld    (b_date),a
        ld    a,(iy+2)
        ld    (b_day),a
        ld    a,(iy+3)
        ld    (b_hour),a
        ld    a,(iy+4)
        ld    (b_min),a
        ld    a,(iy+5)
        ld    (b_sec),a
        call  dis_f3
        ld    a,72
        call  gotoxy
        ld    a,(iy+6)
        ld    (b_mon),a
        ld    a,(iy+7)
        ld    (b_date),a
        ld    a,(iy+8)
        ld    (b_day),a
        ld    a,(iy+9)
        ld    (b_hour),a
        ld    a,(iy+0ah)
        ld    (b_min),a
        ld    a,(iy+0bh)
        ld    (b_sec),a
        call  dis_f3
        pop   af
        ret
edit_6: push  af
        call  clrscr
        call  delay
        ld    a,(s_gl)
        ld    c,a
        inc   c
        ld    b,c
        call  bin_bcd
        ld    c,b
        call  hex_ascii
        call  wrbyte
        ld    d,e
        call  wrbyte
        ld    a,5
        call  gotoxy
        ld    a,(iy+0)
        ld    (b_mon),a
        ld    a,(iy+1)
        ld    (b_date),a
        ld    a,(iy+2)
        ld    (b_day),a
        ld    a,(iy+3)
        ld    (b_hour),a
        ld    a,(iy+4)
        ld    (b_min),a
        ld    a,(iy+5)
        ld    (b_sec),a
        call  dis_f4
        ld    a,69
        call  gotoxy
        ld    a,(iy+6)
        ld    (b_mon),a
        ld    a,(iy+7)
        ld    (b_date),a
        ld    a,(iy+8)
        ld    (b_day),a
        ld    a,(iy+9)
        ld    (b_hour),a
        ld    a,(iy+0ah)
        ld    (b_min),a
        ld    a,(iy+0bh)
        ld    (b_sec),a
        call  dis_f4
        pop   af
```

```
            ret
edit_em:    call  clrscr
            call  delay
            call  off_cursor
            ld    b,1
            ld    c,0
            ld    hl,se_4
            call  print
            call  wait
            call  wait
            call  wait
            jp    edit_e
se_1:       dfb   "    Edit data$"
se_2:       dfb   "Function  $"
se_3:       dfb   "Total      PROGRAM$"
se_4:       dfb   "   PROGRAM  EMPTY$"
chk_pro:    push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  time
            ld    hl,b_status
c_pro1:     ld    a,(b_manual2)
            bit   0,a
            jp    z,c_pro11
            ld    a,(b_manual1)
            bit   0,a
            call  z,chk_off1
            call  nz,chk_on1
            jp    c_pro2
c_pro11:    ld    ix,buff_pro1
            ld    iy,buff_pro1+4
            call  chk_pro1
            cp    1
            call  z,chk_on1
            ld    iy,buff_pro1+4
            call  chk_pro2
            cp    1
            call  z,chk_off1
c_pro2:     ld    a,(b_manual2)
            bit   1,a
            jp    z,c_pro21
            ld    a,(b_manual1)
            bit   1,a
            call  z,chk_off2
            call  nz,chk_on2
            jp    c_pro3
c_pro21:    ld    ix,buff_pro2
            ld    iy,buff_pro2+4
            call  chk_pro1
            cp    1
            call  z,chk_on2
            ld    iy,buff_pro2+4
            call  chk_pro2
            cp    1
            call  z,chk_off2
c_pro3:     ld    a,(b_manual2)
            bit   2,a
            jp    z,c_pro31
            ld    a,(b_manual1)
            bit   2,a
            call  z,chk_off3
            call  nz,chk_on3
            jp    c_pro4
c_pro31:    ld    ix,buff_pro3
            ld    iy,buff_pro3+4
            call  chk_pro1
            cp    1
            call  z,chk_on3
            ld    iy,buff_pro3+4
            call  chk_pro2
            cp    1
            call  z,chk_off3
c_pro4:     ld    a,(b_manual2)
            bit   3,a
            jp    z,c_pro41
            ld    a,(b_manual1)
            bit   3,a
            call  z,chk_off4
            call  nz,chk_on4
            jp    c_pro5
c_pro41:    ld    ix,buff_pro4
            ld    iy,buff_pro4+4
            call  chk_pro1
            cp    1
            call  z,chk_on4
            ld    iy,buff_pro4+4
            call  chk_pro2
            cp    1
            call  z,chk_off4
c_pro5:     ld    a,(b_manual2)
            bit   4,a
            jp    z,c_pro51
            ld    a,(b_manual1)
            bit   4,a
            call  z,chk_off5
            call  nz,chk_on5
            jp    c_pro6
c_pro51:    ld    ix,buff_pro5
            ld    iy,buff_pro5+4
            call  chk_pro1
            cp    1
            call  z,chk_on5
            ld    iy,buff_pro5+4
            call  chk_pro2
            cp    1
            call  z,chk_off5
c_pro6:     ld    a,(b_manual2)
            bit   5,a
            jp    z,c_pro61
            ld    a,(b_manual1)
            bit   5,a
            call  z,chk_off6
            call  nz,chk_on6
            jp    c_pro7
c_pro61:    ld    ix,buff_pro6
            ld    iy,buff_pro6+4
            call  chk_pro1
```

```
            cp    1
            call  z,chk_on6
            ld    iy,buff_pro6+4
            call  chk_pro2
            cp    1
            call  z,chk_off6
c_pro7:     ld    a,(b_manual2)
            bit   6,a
            jp    z,c_pro71
            ld    a,(b_manual1)
            bit   6,a
            call  z,chk_off7
            call  nz,chk_on7
            jp    c_pro8
c_pro71:    ld    ix,buff_pro7
            ld    iy,buff_pro7+4
            call  chk_pro1
            cp    1
            call  z,chk_on7
            ld    iy,buff_pro7+4
            call  chk_pro2
            cp    1
            call  z,chk_off7
c_pro8:     ld    a,(b_manual2)
            bit   7,a
            jp    z,c_pro81
            ld    a,(b_manual1)
            bit   7,a
            call  z,chk_off8
            call  nz,chk_on8
            jp    c_pro9
c_pro81:    ld    ix,buff_pro8
            ld    iy,buff_pro8+4
            call  chk_pro1
            cp    1
            call  z,chk_on8
            ld    iy,buff_pro8+4
            call  chk_pro2
            cp    1
            call  z,chk_off8
c_pro9:     ld    a,(b_status)
            out   (p_load),a
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
chk_on1:    sett  0,(hl)
            ret
chk_on2:    sett  1,(hl)
            ret
chk_on3:    sett  2,(hl)
            ret
chk_on4:    sett  3,(hl)
            ret
chk_on5:    sett  4,(hl)
            ret
chk_on6:    sett  5,(hl)
            ret
chk_on7:    sett  6,(hl)
            ret
chk_on8:    sett  7,(hl)
            ret
chk_off1:   res   0,(hl)
            ret
chk_off2:   res   1,(hl)
            ret
chk_off3:   res   2,(hl)
            ret
chk_off4:   res   3,(hl)
            ret
chk_off5:   res   4,(hl)
            ret
chk_off6:   res   5,(hl)
            ret
chk_off7:   res   6,(hl)
            ret
chk_off8:   res   7,(hl)
            ret
chk_pro1:   ld    a,(ix+0)
            cp    0
            ret   z
            ld    a,(ix+1)
            cp    1
            jp    z,chk_pof1
            cp    2
            jp    z,chk_pof2
            cp    3
            jp    z,chk_pof3
            cp    4
            jp    z,chk_pof4
            ret
chk_pro2:   ld    a,(ix+0)
            cp    0
            ret   z
            ld    a,(ix+1)
            cp    1
            jp    z,chk_prof1
            cp    2
            jp    z,chk_prof2
            cp    3
            jp    z,chk_prof3
            cp    4
            jp    z,chk_prof4
            ret
chk_pof1:   ld    b,(ix+0)
chk_pof1c:  ld    a,(b_hour)
            cp    (iy+3)
            jr    nz,chk_pof11
            ld    a,(b_min)
            cp    (iy+4)
            jr    nz,chk_pof11
            ld    a,(b_sec)
            cp    (iy+5)
            jr    nz,chk_pof11
            ld    a,1
            ret
chk_pof11:  call  inc_ch
```

```
                djnz  chk_pof1c
                ld    a,0
                ret
chk_pof2:       ld    b,(ix+0)
chk_pof2c:      ld    a,(b_day)
                cp    (iy+2)
                jr    nz,chk_pof21
                ld    a,(b_hour)
                cp    (iy+3)
                jr    nz,chk_pof21
                ld    a,(b_min)
                cp    (iy+4)
                jr    nz,chk_pof21
                ld    a,(b_sec)
                cp    (iy+5)
                jr    nz,chk_pof21
                ld    a,1
                ret
chk_pof21:      call  inc_ch
                djnz  chk_pof2c
                ld    a,0
                ret
chk_pof3:       ld    b,(ix+0)
chk_pof3c:      ld    a,(b_date)
                cp    (iy+1)
                jr    nz,chk_pof31
                ld    a,(b_hour)
                cp    (iy+3)
                jr    nz,chk_pof31
                ld    a,(b_min)
                cp    (iy+4)
                jr    nz,chk_pof31
                ld    a,(b_sec)
                cp    (iy+5)
                jr    nz,chk_pof31
                ld    a,1
                ret
chk_pof31:      call  inc_ch
                djnz  chk_pof3c
                ld    a,0
                ret
chk_pof4:       ld    b,(ix+0)
chk_pof4c:      ld    a,(b_mon)
                cp    (iy+0)
                jr    nz,chk_pof41
                ld    a,(b_date)
                cp    (iy+1)
                jr    nz,chk_pof41
                ld    a,(b_hour)
                cp    (iy+3)
                jr    nz,chk_pof41
                ld    a,(b_min)
                cp    (iy+4)
                jr    nz,chk_pof41
                ld    a,(b_sec)
                cp    (iy+5)
                jr    nz,chk_pof41
                ld    a,1
                ret
chk_pof41:      call  inc_ch
                djnz  chk_pof4c
                ld    a,0
                ret
chk_prof1:      ld    b,(ix+0)
chk_prof1c:     ld    a,(b_hour)
                cp    (iy+9)
                jr    nz,chk_prof11
                ld    a,(b_min)
                cp    (iy+10)
                jr    nz,chk_prof11
                ld    a,(b_sec)
                cp    (iy+11)
                jr    nz,chk_prof11
                ld    a,1
                ret
chk_prof11:     call  inc_ch
                djnz  chk_prof1c
                ld    a,0
                ret
chk_prof2:      ld    b,(ix+0)
chk_prof2c:     ld    a,(b_day)
                cp    (iy+8)
                jr    nz,chk_prof21
                ld    a,(b_hour)
                cp    (iy+9)
                jr    nz,chk_prof21
                ld    a,(b_min)
                cp    (iy+10)
                jr    nz,chk_prof21
                ld    a,(b_sec)
                cp    (iy+11)
                jr    nz,chk_prof21
                ld    a,1
                ret
chk_prof21:     call  inc_ch
                djnz  chk_prof2c
                ld    a,0
                ret
chk_prof3:      ld    b,(ix+0)
chk_prof3c:     ld    a,(b_date)
                cp    (iy+7)
                jr    nz,chk_prof31
                ld    a,(b_hour)
                cp    (iy+9)
                jr    nz,chk_prof31
                ld    a,(b_min)
                cp    (iy+10)
                jr    nz,chk_prof31
                ld    a,(b_sec)
                cp    (iy+11)
                jr    nz,chk_prof31
                ld    a,1
                ret
chk_prof31:     call  inc_ch
                djnz  chk_prof3c
                ld    a,0
                ret
chk_prof4:      ld    b,(ix+0)
chk_prof4c:     ld    a,(b_mon)
                cp    (iy+6)
```

```
            jr    nz,chk_prof41
            ld    a,(b_date)
            cp    (iy+7)
            jr    nz,chk_prof41
            ld    a,(b_hour)
            cp    (iy+9)
            jr    nz,chk_prof41
            ld    a,(b_min)
            cp    (iy+10)
            jr    nz,chk_prof41
            ld    a,(b_sec)
            cp    (iy+11)
            jr    nz,chk_prof41
            ld    a,1
            ret
chk_prof41: call  inc_ch
            djnz  chk_prof4c
            ld    a,0
            ret
;************************
;*  Display status LOAD  *
;************************
status:     push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            ld    ix,dis_st
            ld    (ix+0),31h
            ld    (ix+1),32h
            ld    (ix+2),33h
            ld    (ix+3),34h
            ld    (ix+4),35h
            ld    (ix+5),36h
            ld    (ix+6),37h
            ld    (ix+7),38h
            ld    (ix+8),"$"
            ld    iy,b_status
            ld    c,(iy+0)
            bit   0,c
            call  nz,sta_0
            inc   ix
            bit   1,c
            call  nz,sta_0
            inc   ix
            bit   2,c
            call  nz,sta_0
            inc   ix
            bit   3,c
            call  nz,sta_0
            inc   ix
            bit   4,c
            call  nz,sta_0
            inc   ix
            bit   5,c
            call  nz,sta_0
            inc   ix
            bit   6,c
            call  nz,sta_0
            inc   ix
            bit   7,c
            call  nz,sta_0
            ld    c,(iy+2)
            ld    ix,dis_st
            bit   0,c
            call  nz,status_1
            inc   ix
            bit   1,c
            call  nz,status_2
            inc   ix
            bit   2,c
            call  nz,status_3
            inc   ix
            bit   3,c
            call  nz,status_4
            inc   ix
            bit   4,c
            call  nz,status_5
            inc   ix
            bit   5,c
            call  nz,status_6
            inc   ix
            bit   6,c
            call  nz,status_7
            inc   ix
            bit   7,c
            call  nz,status_8
            ld    b,2
            ld    c,3
            ld    hl,st_s1
            call  print
            ld    b,2
            ld    c,9
            ld    hl,dis_st
            call  print
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
status_1:   ld    b,(iy+1)
            bit   0,b
            call  z,sta_1
            call  nz,sta_2
            ret
status_2:   ld    b,(iy+1)
            bit   1,b
            call  z,sta_1
            call  nz,sta_2
            ret
status_3:   ld    b,(iy+1)
            bit   2,b
            call  z,sta_1
            call  nz,sta_2
            ret
status_4:   ld    b,(iy+1)
            bit   3,b
```

```
            call  z,sta_1
            call  nz,sta_2
            ret
status_5:   ld    b,(iy+1)
            bit   4,b
            call  z,sta_1
            call  nz,sta_2
            ret
status_6:   ld    b,(iy+1)
            bit   5,b
            call  z,sta_1
            call  nz,sta_2
            ret
status_7:   ld    b,(iy+1)
            bit   6,b
            call  z,sta_1
            call  nz,sta_2
            ret
status_8:   ld    b,(iy+1)
            bit   7,b
            call  z,sta_1
            call  nz,sta_2
            ret
sta_0:      ld    (ix+0),05fh
            ret
sta_1:      ld    (ix+0),0ffh
            ret
sta_2:      ld    (ix+0),"o"
            ret
st_s1:      dfb   "LOAD $"
;*******************************
;*  Select channel             *
;*  Input  = -                 *
;*  Output = IX  total         *
;*           IY  data program  *
;*******************************
select_ch:  push  af
            push  bc
            push  de
            push  hl
            ld    ix,s_g1
            ld    a,0
            ld    (ix+0),a
            ld    (s_g6),a
            ld    b,1
            ld    c,0
            ld    hl,st_p1
            call  print
            ld    a,9
            call  gotoxy
            call  delay
            call  on_cursor
            call  delay
            ld    c,1
program1:   call  scan_key
            cp    k_left
            call  z,program2
            cp    k_right
            call  z,program3
            cp    k_main
            jp    z,sel_ch
            cp    k_enter
            jr    z,program4
            jr    program1
program4:   ld    a,c
            ld    (s_g1),a
            call  hex_ascii
            ld    a,9
            call  gotoxy
            ld    d,e
            call  wrbyte
            ld    d,20h
            call  wrbyte
            call  wrbyte
            call  wrbyte
            call  wrbyte
            call  wrbyte
            call  wrbyte
            call  wrbyte
            call  wrbyte
            ld    a,(s_g1)
            cp    1
            call  z,program71
            cp    2
            call  z,program72
            cp    3
            call  z,program73
            cp    4
            call  z,program74
            cp    5
            call  z,program75
            cp    6
            call  z,program76
            cp    7
            call  z,program77
            cp    8
            call  z,program78
            call  wait
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
sel_ch:     ld    a,1
            ld    (s_g6),a
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
program71:  ld    ix,buff_pro1
            ld    iy,buff_pro1+4
            ret
program72:  ld    ix,buff_pro2
            ld    iy,buff_pro2+4
            ret
program73:  ld    ix,buff_pro3
            ld    iy,buff_pro3+4
            ret
program74:  ld    ix,buff_pro4
            ld    iy,buff_pro4+4
```

```
            ret
program75:  ld    ix,buff_pro5
            ld    iy,buff_pro5+4
            ret
program76:  ld    ix,buff_pro6
            ld    iy,buff_pro6+4
            ret
program77:  ld    ix,buff_pro7
            ld    iy,buff_pro7+4
            ret
program78:  ld    ix,buff_pro8
            ld    iy,buff_pro8+4
            ret
;********************
;*  Clear program  *
;********************
clear_pro:  push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  delay
            call  clrscr
            call  delay
            ld    b,2
            ld    c,0
            ld    hl,st_c1
            call  print
            call  select_ch
            ld    a,(s_g1)
            cp    0
            jr    z,cl_end
            call  clrscr
            ld    b,1
            ld    c,0
            ld    hl,st_c2
            call  print
            ld    c,a
            call  hex_ascii
            call  wrbyte
            call  hex_ascii
            ld    d,e
            call  wrbyte
            ld    b,2
            ld    c,0
            ld    hl,st_c3
            call  print
            ld    a,0
            ld    (ix+0),a
            call  wait
            call  wait
            call  wait
cl_end:     call  clrscr
            call  delay
            call  off_cursor
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
;******************************
;*  Program load ON/OFF load  *
;******************************
program:    push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  delay
            call  clrscr
            call  delay
            call  select_ch
            ld    a,(s_g6)
            cp    1
            jp    z,program_e
            ld    a,(ix+0)
            cp    0
            jp    nz,pro_f
            ld    b,2
            ld    c,0
            ld    hl,st_p2
            call  print
            jr    pro_ff5
pro_f:      ld    b,a
pro_ff:     push  bc
            ld    b,12
pro_ff1:    inc   iy
            djnz  pro_ff1
            pop   bc
            djnz  pro_ff
            jp    pro_f1
pro_ff5:    ld    a,49h
            call  gotoxy
            ld    c,1
program41:  call  scan_key
            cp    k_left
            call  z,program5
            cp    k_right
            call  z,program6
            cp    k_main
            jp    z,program_e
            cp    k_enter
            jr    z,program7
            jr    program41
program7:   ld    a,c
            ld    (ix+1),a
            call  hex_ascii
            ld    a,49h
            call  gotoxy
            ld    d,e
            call  wrbyte
            ld    d,20h
            call  wrbyte
            call  wrbyte
            call  wrbyte
            call  wrbyte
```

```
            call  wrbyte
            call  wrbyte
            call  wrbyte
            call  off_cursor
            call  wait
            call  wait
            call  wait
pro_f1:     call  delay
            call  clrscr
            call  delay
            call  on_cursor
            ld    b,1
            ld    c,0
            ld    hl,st_p3
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_p4
            call  print
            ld    a,8
            call  gotoxy
            ld    a,(ix+0)
            ld    b,a
            call  bin_bcd
            ld    c,b
            call  hex_ascii
            call  wrbyte
            ld    d,e
            call  wrbyte
            ld    a,52h
            call  gotoxy
prog_loop:  call  scan_key
            cp    k_main
            jp    z,program_e
            cp    k_enter
            jr    z,program8
            jr    prog_loop
program8:   call  cl_b
            call  wait
            ld    a,(ix+1)
            cp    1
            jp    z,program81
            cp    2
            jp    z,program82
            cp    3
            jp    z,program83
            cp    4
            jp    z,program84
program81:  call  clrscr
            call  delay
            ld    b,1
            ld    c,0
            ld    hl,st_pf1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_pf11
            call  print
            ld    a,11
            call  gotoxy
            call  dis_f1
            ld    a,11
            call  gotoxy
            ld    c,1
program811: call  ta_p1
            call  scan_key
            cp    k_left
            call  z,p1_left
            cp    k_right
            call  z,p1_right
            cp    k_inc
            call  z,p1_inc
            cp    k_dec
            call  z,p1_dec
            cp    k_main
            jp    z,program_e
            cp    k_enter
            jp    z,program812
            call  key_press
            jr    program811
program812: call  update
            call  wait
            ld    a,(s_g6)
            cp    1
            jp    z,program81
            ld    a,(b_mon)
            ld    (iy+0),a
            ld    a,(b_date)
            ld    (iy+1),a
            ld    a,(b_day)
            ld    (iy+2),a
            ld    a,(b_hour)
            ld    (iy+3),a
            ld    a,(b_min)
            ld    (iy+4),a
            ld    a,(b_sec)
            ld    (iy+5),a
up1:        ld    a,75
            call  gotoxy
            call  dis_f1
            ld    a,11
            call  gotoxy
            ld    c,1
program813: call  ota_p1
            call  scan_key
            cp    k_left
            call  z,p1_left
            cp    k_right
            call  z,p1_right
            cp    k_inc
            call  z,op1_inc
            cp    k_dec
            call  z,op1_dec
            cp    k_main
            jp    z,program_e
            cp    k_enter
            jp    z,program814
            call  key1_press
            jr    program813
program814: call  update
```

```
                ld    a,(s_g6)
                cp    1
                jp    z,up1
                call  no_update
                ld    a,(s_g6)
                cp    1
                jp    z,pro_f1
                ld    a,(b_mon)
                ld    (iy+6),a
                ld    a,(b_date)
                ld    (iy+7),a
                ld    a,(b_day)
                ld    (iy+8),a
                ld    a,(b_hour)
                ld    (iy+9),a
                ld    a,(b_min)
                ld    (iy+10),a
                ld    a,(b_sec)
                ld    (iy+11),a
                call  inc_ch
                inc   (ix+0)
                jp    pro_f1
pl_key1:        ld    a,(hl)
                and   0fh
                ld    (hl),a
                ld    a,c
                rlca
                rlca
                rlca
                rlca
                or    (hl)
                ld    (hl),a
                ret
pl_key2:        ld    a,(hl)
                and   0f0h
                ld    (hl),a
                ld    a,c
                or    (hl)
                ld    (hl),a
                ret
key_press:      push  bc
                call  chang_key
                ld    a,(s_g5)
                cp    0
                jr    z,key_pre_e
                ld    a,e
                cp    1
                call  z,pl_key1
                cp    2
                call  z,pl_key2
                ld    a,11
                call  gotoxy
                call  dis_f1
                pop   bc
                ld    a,c
                cp    6
                jr    z,key_pre_e1
                inc   c
                ret
key_pre_e:      pop   bc
key_pre_e1:     ret
key1_press:     push  bc
                call  chang_key
                ld    a,(s_g5)
                cp    0
                jr    z,key1_pe
                ld    a,e
                cp    1
                call  z,pl_key1
                cp    2
                call  z,pl_key2
                ld    a,75
                call  gotoxy
                call  dis_f1
                pop   bc
                ld    a,c
                cp    6
                jr    z,key1_pe1
                inc   c
                ret
key1_pe:        pop   bc
key1_pe1:       ret
cl_b:           push  af
                push  bc
                push  de
                push  hl
                ld    a,1
                ld    (b_mon),a
                ld    (b_date),a
                ld    (b_day),a
                ld    a,0
                ld    (b_hour),a
                ld    (b_min),a
                ld    (b_sec),a
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
dis_f1:         push  af
                push  bc
                push  de
                push  hl
                ld    hl,b_hour
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
                call  wrbyte
                ld    hl,b_min
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
```

```
            call  wrbyte
            ld    hl,b_sec
            ld    c,(hl)
            call  hex_ascii
            call  wrbyte
            call  hex_ascii
            ld    d,e
            call  wrbyte
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
pl_inc:     push  bc
            ld    a,e
            cp    1
            call  z,pl_inc1
            cp    2
            call  z,pl_inc2
            pop   bc
            ld    a,11
            call  gotoxy
            call  dis_f1
            ret
pl_dec:     push  bc
            ld    a,e
            cp    1
            call  z,pl_dec1
            cp    2
            call  z,pl_dec2
            pop   bc
            ld    a,11
            call  gotoxy
            call  dis_f1
            ret
op1_inc:    push  bc
            ld    a,e
            cp    1
            call  z,pl_inc1
            cp    2
            call  z,pl_inc2
            pop   bc
            ld    a,75
            call  gotoxy
            call  dis_f1
            ret
op1_dec:    push  bc
            ld    a,e
            cp    1
            call  z,pl_dec1
            cp    2
            call  z,pl_dec2
            pop   bc
            ld    a,75
            call  gotoxy
            call  dis_f1
            ret
pl_inc1:    ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    b
            call  z,pl_inc10
            inc   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
pl_inc10:   ld    a,d
            dec   a
            ret
pl_inc2:    ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    b
            call  z,pl_inc10
            inc   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
pl_dec1:    ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    d
            call  z,pl_dec10
            dec   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
pl_dec10:   ld    a,b
            inc   a
```

```
            ret
pl_dec2:    ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    d
            call  z,pl_dec10
            dec   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
pl_left:    dec   c
            ld    a,c
            cp    0
            call  z,pl_left_d
            ret
pl_left_d:  ld    c,6
            ret
pl_right:   inc   c
            ld    a,c
            cp    7
            call  z,pl_left_t
            ret
pl_left_t:  ld    c,1
            ret
ota_pl:     push  ix
            ld    ix,otbl_1
            jr    table_pl
ta_pl:      push  ix
            ld    ix,tbl_1
table_pl:   ld    hl,b_hour
            dec   ix
            dec   ix
            dec   ix
            dec   ix
            inc   hl
            ld    b,c
tbl_11:     inc   ix
            inc   ix
            inc   ix
            inc   ix
            djnz  tbl_11
            ld    a,(ix+0)
            call  gotoxy
            ld    d,(ix+1)
            ld    b,(ix+2)
            ld    e,(ix+3)
            ld    a,c
            cp    1
            jr    z,tbl_111
            cp    2
            jr    z,tbl_111
            cp    3
            jr    z,tbl_112
            cp    4
            jr    z,tbl_112
            cp    5
            jr    z,tbl_113
            cp    6
            jr    z,tbl_113
tbl_113:    dec   hl
tbl_112:    dec   hl
tbl_111:    dec   hl
            pop   ix
            ret
tbl_1:      dfb   11,0,2,1
tbl_2:      dfb   12,0,9,2
tbl_3:      dfb   14,0,5,1
tbl_4:      dfb   15,0,9,2
tbl_5:      dfb   17,0,5,1
tbl_6:      dfb   18,0,9,2
otbl_1:     dfb   75,0,2,1
otbl_2:     dfb   76,0,9,2
otbl_3:     dfb   78,0,5,1
otbl_4:     dfb   79,0,9,2
otbl_5:     dfb   81,0,5,1
otbl_6:     dfb   82,0,9,2
program82:  call  clrscr
            call  delay
            ld    b,1
            ld    c,0
            ld    hl,st_pf1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_pf11
            call  print
            ld    a,7
            call  gotoxy
            call  dis_f2
            ld    a,7
            call  gotoxy
            ld    c,1
program821: call  ta_p2
            call  scan_key
            cp    k_left
            call  z,p2_left
            cp    k_right
            call  z,p2_right
            cp    k_inc
            call  z,p2_inc
            cp    k_dec
            call  z,p2_dec
            cp    k_main
            jp    z,program_e
            cp    k_enter
            jp    z,program822
            call  key_press2
            jr    program821
program822: call  update
            call  wait
            ld    a,(s_g6)
            cp    1
            jp    z,program82
            ld    a,(b_mon)
            ld    (iy+0),a
```

```
                ld    a,(b_date)
                ld    (iy+1),a
                ld    a,(b_day)
                ld    (iy+2),a
                ld    a,(b_hour)
                ld    (iy+3),a
                ld    a,(b_min)
                ld    (iy+4),a
                ld    a,(b_sec)
                ld    (iy+5),a
up2:            ld    a,71
                call  gotoxy
                call  dis_f2
                ld    a,71
                call  gotoxy
                ld    c,1
program823:     call  ota_p2
                call  scan_key
                cp    k_left
                call  z,p2_left
                cp    k_right
                call  z,p2_right
                cp    k_inc
                call  z,op2_inc
                cp    k_dec
                call  z,op2_dec
                cp    k_main
                jp    z,program_e
                cp    k_enter
                jp    z,program824
                call  key1_p2
                jr    program823
program824:     call  update
                ld    a,(s_g6)
                cp    1
                jp    z,up2
                call  no_update
                ld    a,(s_g6)
                cp    1
                jp    z,pro_f1
                ld    a,(b_mon)
                ld    (iy+6),a
                ld    a,(b_date)
                ld    (iy+7),a
                ld    a,(b_day)
                ld    (iy+8),a
                ld    a,(b_hour)
                ld    (iy+9),a
                ld    a,(b_min)
                ld    (iy+10),a
                ld    a,(b_sec)
                ld    (iy+11),a
                call  inc_ch
                inc   (ix+0)
                jp    pro_f1
p2_key1:        ld    a,(hl)
                and   0fh
                ld    (hl),a
                ld    a,c
                rlca
                rlca
                rlca
                or    (hl)
                ld    (hl),a
                ret
p2_key2:        ld    a,(hl)
                and   0f0h
                ld    (hl),a
                ld    a,c
                or    (hl)
                ld    (hl),a
                ret
key_press2:     push  bc
                push  af
                ld    a,c
                cp    1
                jr    z,k2_pe1
                pop   af
                call  chang_key
                ld    a,(s_g5)
                cp    0
                jr    z,k2_pe
                ld    a,e
                cp    1
                call  z,p2_key1
                cp    2
                call  z,p2_key2
                ld    a,7
                call  gotoxy
                call  dis_f2
                pop   bc
                ld    a,c
                cp    7
                jr    z,k2_pre_e1
                inc   c
                ret
k2_pe1:         pop   af
k2_pe:          pop   bc
k2_pre_e1:      ret
key1_p2:        push  bc
                push  af
                ld    a,c
                cp    1
                jr    z,k2_pe1
                pop   af
                call  chang_key
                ld    a,(s_g5)
                cp    0
                jr    z,k2_pe
                ld    a,e
                cp    1
                call  z,p2_key1
                cp    2
                call  z,p2_key2
                ld    a,71
                call  gotoxy
                call  dis_f2
                pop   bc
                ld    a,c
```

```
                cp    7
                jr    z,k2_pre_el
                inc   c
                ret
dis_f2:         push  af
                push  bc
                push  de
                push  hl
                ld    hl,b_day
                ld    a,(hl)
                cp    1
                call  z,dis_f21
                cp    2
                call  z,dis_f22
                cp    3
                call  z,dis_f23
                cp    4
                call  z,dis_f24
                cp    5
                call  z,dis_f25
                cp    6
                call  z,dis_f26
                cp    7
                call  z,dis_f27
                ld    d,(hl)
                call  wrbyte
                inc   hl
                ld    d,(hl)
                call  wrbyte
                inc   hl
                ld    d,(hl)
                call  wrbyte
                ld    d,20h
                call  wrbyte
                ld    hl,b_hour
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
                call  wrbyte
                ld    hl,b_min
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
                call  wrbyte
                ld    hl,b_sec
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
dis_f21:        ld    hl,sun
                ret
dis_f22:        ld    hl,mon
                ret
dis_f23:        ld    hl,tue
                ret
dis_f24:        ld    hl,wed
                ret
dis_f25:        ld    hl,thu
                ret
dis_f26:        ld    hl,fri
                ret
dis_f27:        ld    hl,sat
                ret
p2_inc:         push  bc
                ld    a,e
                cp    1
                call  z,p2_incl
                cp    2
                call  z,p2_inc2
                pop   bc
                ld    a,7
                call  gotoxy
                call  dis_f2
                ret
p2_dec:         push  bc
                ld    a,e
                cp    1
                call  z,p2_decl
                cp    2
                call  z,p2_dec2
                pop   bc
                ld    a,7
                call  gotoxy
                call  dis_f2
                ret
op2_inc:        push  bc
                ld    a,e
                cp    1
                call  z,p2_incl
                cp    2
                call  z,p2_inc2
                pop   bc
                ld    a,71
                call  gotoxy
                call  dis_f2
                ret
op2_dec:        push  bc
                ld    a,e
                cp    1
                call  z,p2_decl
                cp    2
                call  z,p2_dec2
                pop   bc
                ld    a,71
                call  gotoxy
```

```
            call  dis_f2
            ret
p2_incl:    ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    b
            call  z,p2_incl0
            inc   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p2_incl0:   ld    a,d
            dec   a
            ret
p2_inc2:    ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    b
            call  z,p2_incl0
            inc   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p2_decl:    ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    d
            call  z,p2_decl0
            dec   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p2_decl0:   ld    a,b
            inc   a
            ret
p2_dec2:    ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    d
            call  z,p2_decl0
            dec   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p2_left:    dec   c
            ld    a,c
            cp    0
            call  z,p2_left_d
            ret
p2_left_d:  ld    c,7
            ret
p2_right:   inc   c
            ld    a,c
            cp    8
            call  z,p2_left_t
            ret
p2_left_t:  ld    c,1
            ret
ota_p2:     push  ix
            ld    a,c
            cp    1
            call  z,ot2_tbl1
            cp    2
            call  z,ot2_tbl2
            cp    3
            call  z,ot2_tbl3
            cp    4
            call  z,ot2_tbl4
            cp    5
            call  z,ot2_tbl5
            cp    6
            call  z,ot2_tbl6
            cp    7
            call  z,ot2_tbl7
            ld    a,(ix+0)
            call  gotoxy
            ld    d,(ix+1)
            ld    b,(ix+2)
            ld    e,(ix+3)
            pop   ix
            ret
ot2_tbl1:   ld    ix,otbl2_1
            ld    hl,b_day
            ret
```

```
ot2_tbl2:   ld    ix,otbl2_2
            ld    hl,b_hour
            ret
ot2_tbl3:   ld    ix,otbl2_3
            ld    hl,b_hour
            ret
ot2_tbl4:   ld    ix,otbl2_4
            ld    hl,b_min
            ret
ot2_tbl5:   ld    ix,otbl2_5
            ld    hl,b_min
            ret
ot2_tbl6:   ld    ix,otbl2_6
            ld    hl,b_sec
            ret
ot2_tbl7:   ld    ix,otbl2_7
            ld    hl,b_sec
            ret
ta_p2:      push  ix
table_p2:   ld    a,c
            cp    1
            call  z,t2_tbl1
            cp    2
            call  z,t2_tbl2
            cp    3
            call  z,t2_tbl3
            cp    4
            call  z,t2_tbl4
            cp    5
            call  z,t2_tbl5
            cp    6
            call  z,t2_tbl6
            cp    7
            call  z,t2_tbl7
            ld    a,(ix+0)
            call  gotoxy
            ld    d,(ix+1)
            ld    b,(ix+2)
            ld    e,(ix+3)
            pop   ix
            ret
t2_tbl1:    ld    ix,tbl2_1
            ld    hl,b_day
            ret
t2_tbl2:    ld    ix,tbl2_2
            ld    hl,b_hour
            ret
t2_tbl3:    ld    ix,tbl2_3
            ld    hl,b_hour
            ret
t2_tbl4:    ld    ix,tbl2_4
            ld    hl,b_min
            ret
t2_tbl5:    ld    ix,tbl2_5
            ld    hl,b_min
            ret
t2_tbl6:    ld    ix,tbl2_6
            ld    hl,b_sec
            ret
t2_tbl7:    ld    ix,tbl2_7
            ld    hl,b_sec
            ret
tbl2_1:     dfb   7,1,7,2
tbl2_2:     dfb   11,0,2,1
tbl2_3:     dfb   12,0,9,2
tbl2_4:     dfb   14,0,5,1
tbl2_5:     dfb   15,0,9,2
tbl2_6:     dfb   17,0,5,1
tbl2_7:     dfb   18,0,9,2
otbl2_1:    dfb   71,1,7,2
otbl2_2:    dfb   75,0,2,1
otbl2_3:    dfb   76,0,9,2
otbl2_4:    dfb   78,0,5,1
otbl2_5:    dfb   79,0,9,2
otbl2_6:    dfb   81,0,5,1
otbl2_7:    dfb   82,0,9,2
program83:  call  clrscr
            call  delay
            ld    b,1
            ld    c,0
            ld    hl,st_pf1
            call  print
            ld    b,2
            ld    c,0
            ld    hl,st_pf11
            call  print
            ld    a,8
            call  gotoxy
            call  dis_f3
            ld    c,1
program831: call  ta_p3
            call  scan_key
            cp    k_left
            call  z,p3_left
            cp    k_right
            call  z,p3_right
            cp    k_inc
            call  z,p3_inc
            cp    k_dec
            call  z,p3_dec
            cp    k_main
            jp    z,program_e
            cp    k_enter
            jp    z,program832
            call  key_press3
            jr    program831
program832: call  update
            call  wait
            ld    a,(s_g6)
            cp    1
            jp    z,program83
            ld    a,(b_mon)
            ld    (iy+0),a
            ld    a,(b_date)
            ld    (iy+1),a
            ld    a,(b_day)
            ld    (iy+2),a
            ld    a,(b_hour)
            ld    (iy+3),a
            ld    a,(b_min)
```

```
            ld    (iy+4),a
            ld    a,(b_sec)
            ld    (iy+5),a
up3:        ld    a,72
            call  gotoxy
            call  dis_f3
            ld    a,8
            call  gotoxy
            ld    c,1
program833: call  ota_p3
            call  scan_key
            cp    k_left
            call  z,p3_left
            cp    k_right
            call  z,p3_right
            cp    k_inc
            call  z,op3_inc
            cp    k_dec
            call  z,op3_dec
            cp    k_main
            jp    z,program_e
            cp    k_enter
            jp    z,program834
            call  key3_p3
            jr    program833
program834: call  update
            ld    a,(s_g6)
            cp    1
            jp    z,up3
            call  no_update
            ld    a,(s_g6)
            cp    1
            jp    z,pro_f1
            ld    a,(b_mon)
            ld    (iy+6),a
            ld    a,(b_date)
            ld    (iy+7),a
            ld    a,(b_day)
            ld    (iy+8),a
            ld    a,(b_hour)
            ld    (iy+9),a
            ld    a,(b_min)
            ld    (iy+10),a
            ld    a,(b_sec)
            ld    (iy+11),a
            call  inc_ch
            inc   (ix+0)
            jp    pro_f1
p3_key1:    ld    a,(hl)
            and   0fh
            ld    (hl),a
            ld    a,c
            rlca
            rlca
            rlca
            rlca
            or    (hl)
            ld    (hl),a
            ret
p3_key2:    ld    a,(hl)
            and   0f0h
            ld    (hl),a
            ld    a,c
            or    (hl)
            ld    (hl),a
            ret
key_press3: push  bc
            call  chang_key
            ld    a,(s_g5)
            cp    0
            jr    z,k3_pre_e
            ld    a,e
            cp    1
            call  z,p3_key1
            cp    2
            call  z,p3_key2
            ld    a,8
            call  gotoxy
            call  dis_f3
            pop   bc
            ld    a,c
            cp    8
            jr    z,k3_pre_e1
            inc   c
            ret
k3_pre_e:   pop   bc
k3_pre_e1:  ret
key3_p3:    push  bc
            call  chang_key
            ld    a,(s_g5)
            cp    0
            jr    z,k3_pre_e
            ld    a,e
            cp    1
            call  z,p3_key1
            cp    2
            call  z,p3_key2
            ld    a,72
            call  gotoxy
            call  dis_f3
            pop   bc
            ld    a,c
            cp    8
            jr    z,k3_pre_e1
            inc   c
            ret
dis_f3:     push  af
            push  bc
            push  de
            push  hl
            ld    hl,b_date
            ld    c,(hl)
            call  hex_ascii
            call  wrbyte
            call  hex_ascii
            ld    d,e
            call  wrbyte
            ld    d,20h
            call  wrbyte
            ld    hl,b_hour
```

```
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
                call  wrbyte
                ld    hl,b_min
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
                call  wrbyte
                ld    hl,b_sec
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
p3_inc:         push  bc
                ld    a,e
                cp    1
                call  z,p3_inc1
                cp    2
                call  z,p3_inc2
                pop   bc
                ld    a,8
                call  gotoxy
                call  dis_f3
                ret
p3_dec:         push  bc
                ld    a,e
                cp    1
                call  z,p3_dec1
                cp    2
                call  z,p3_dec2
                pop   bc
                ld    a,8
                call  gotoxy
                call  dis_f3
                ret
op3_inc:        push  bc
                ld    a,e
                cp    1
                call  z,p3_inc1
                cp    2
                call  z,p3_inc2
                pop   bc
                ld    a,72
                call  gotoxy
                call  dis_f3
                ret
op3_dec:        push  bc
                ld    a,e
                cp    1
                call  z,p3_dec1
                cp    2
                call  z,p3_dec2
                pop   bc
                ld    a,72
                call  gotoxy
                call  dis_f3
                ret
p3_inc1:        ld    a,(hl)
                and   0fh
                ld    (b_set_time),a
                ld    a,(hl)
                and   0f0h
                rrca
                rrca
                rrca
                rrca
                cp    b
                call  z,p3_inc10
                inc   a
                rlca
                rlca
                rlca
                rlca
                push  hl
                ld    hl,b_set_time
                or    (hl)
                pop   hl
                ld    (hl),a
                ret
p3_inc10:       ld    a,d
                dec   a
                ret
p3_inc2:        ld    a,(hl)
                and   0f0h
                ld    (b_set_time),a
                ld    a,(hl)
                and   0fh
                cp    b
                call  z,p3_inc10
                inc   a
                push  hl
                ld    hl,b_set_time
                or    (hl)
                pop   hl
                ld    (hl),a
                ret
p3_dec1:        ld    a,(hl)
                and   0fh
                ld    (b_set_time),a
                ld    a,(hl)
                and   0f0h
                rrca
                rrca
                rrca
```

```
            rrca
            cp    d
            call  z,p3_dec10
            dec   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p3_dec10:   ld    a,b
            inc   a
            ret
p3_dec2:    ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    d
            call  z,p3_dec10
            dec   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p3_left:    dec   c
            ld    a,c
            cp    0
            call  z,p3_left_d
            ret
p3_left_d:  ld    c,8
            ret
p3_right:   inc   c
            ld    a,c
            cp    9
            call  z,p3_left_t
            ret
p3_left_t:  ld    c,1
            ret
ota_p3:     push  ix
            ld    a,c
            cp    1
            call  z,ot3_tbl1
            cp    2
            call  z,ot3_tbl2
            cp    3
            call  z,ot3_tbl3
            cp    4
            call  z,ot3_tbl4
            cp    5
            call  z,ot3_tbl5
            cp    6
            call  z,ot3_tbl6
            cp    7
            call  z,ot3_tbl7
            cp    8
            call  z,ot3_tbl8
            ld    a,(ix+0)
            call  gotoxy
            ld    d,(ix+1)
            ld    b,(ix+2)
            ld    e,(ix+3)
            pop   ix
            ret
ot3_tbl1:   ld    ix,otbl3_1
            ld    hl,b_date
            ret
ot3_tbl2:   ld    ix,otbl3_2
            ld    hl,b_date
            ret
ot3_tbl3:   ld    ix,otbl3_3
            ld    hl,b_hour
            ret
ot3_tbl4:   ld    ix,otbl3_4
            ld    hl,b_hour
            ret
ot3_tbl5:   ld    ix,otbl3_5
            ld    hl,b_min
            ret
ot3_tbl6:   ld    ix,otbl3_6
            ld    hl,b_min
            ret
ot3_tbl7:   ld    ix,otbl3_7
            ld    hl,b_sec
            ret
ot3_tbl8:   ld    ix,otbl3_8
            ld    hl,b_sec
            ret
ta_p3:      push  ix
table_p3:   ld    a,c
            cp    1
            call  z,t3_tbl1
            cp    2
            call  z,t3_tbl2
            cp    3
            call  z,t3_tbl3
            cp    4
            call  z,t3_tbl4
            cp    5
            call  z,t3_tbl5
            cp    6
            call  z,t3_tbl6
            cp    7
            call  z,t3_tbl7
            cp    8
            call  z,t3_tbl8
            ld    a,(ix+0)
            call  gotoxy
            ld    d,(ix+1)
            ld    b,(ix+2)
            ld    e,(ix+3)
            pop   ix
            ret
t3_tbl1:    ld    ix,tbl3_1
```

```
                ld    hl,b_date
                ret
t3_tbl2:        ld    ix,tbl3_2
                ld    hl,b_date
                ret
t3_tbl3:        ld    ix,tbl3_3
                ld    hl,b_hour
                ret
t3_tbl4:        ld    ix,tbl3_4
                ld    hl,b_hour
                ret
t3_tbl5:        ld    ix,tbl3_5
                ld    hl,b_min
                ret
t3_tbl6:        ld    ix,tbl3_6
                ld    hl,b_min
                ret
t3_tbl7:        ld    ix,tbl3_7
                ld    hl,b_sec
                ret
t3_tbl8:        ld    ix,tbl3_8
                ld    hl,b_sec
                ret
tbl3_1:         dfb   8,0,3,1
tbl3_2:         dfb   9,0,9,2
tbl3_3:         dfb   11,0,2,1
tbl3_4:         dfb   12,0,9,2
tbl3_5:         dfb   14,0,5,1
tbl3_6:         dfb   15,0,9,2
tbl3_7:         dfb   17,0,5,1
tbl3_8:         dfb   18,0,9,2
otbl3_1:        dfb   72,0,3,1
otbl3_2:        dfb   73,0,9,2
otbl3_3:        dfb   75,0,2,1
otbl3_4:        dfb   76,0,9,2
otbl3_5:        dfb   78,0,5,1
otbl3_6:        dfb   79,0,9,2
otbl3_7:        dfb   81,0,5,1
otbl3_8:        dfb   82,0,9,2
program84:      call  clrscr
                call  delay
                ld    b,1
                ld    c,0
                ld    hl,st_pf1
                call  print
                ld    b,2
                ld    c,0
                ld    hl,st_pf11
                call  print
                ld    a,7
                call  gotoxy
                ld    a,5
                call  gotoxy
                call  dis_f4
                ld    c,1
program841:     call  ta_p4
                call  scan_key
                cp    k_left
                call  z,p4_left
                cp    k_right
                call  z,p4_right
                cp    k_inc
                call  z,p4_inc
                cp    k_dec
                call  z,p4_dec
                cp    k_main
                jp    z,program_e
                cp    k_enter
                jp    z,program842
                call  key_press4
                jr    program841
program842:     call  update
                call  wait
                ld    a,(s_g6)
                cp    1
                jp    z,program84
                ld    a,(b_mon)
                ld    (iy+0),a
                ld    a,(b_date)
                ld    (iy+1),a
                ld    a,(b_day)
                ld    (iy+2),a
                ld    a,(b_hour)
                ld    (iy+3),a
                ld    a,(b_min)
                ld    (iy+4),a
                ld    a,(b_sec)
                ld    (iy+5),a
up4:            ld    a,69
                call  gotoxy
                call  dis_f4
                ld    a,8
                call  gotoxy
                ld    c,1
program843:     call  ota_p4
                call  scan_key
                cp    k_left
                call  z,p4_left
                cp    k_right
                call  z,p4_right
                cp    k_inc
                call  z,op4_inc
                cp    k_dec
                call  z,op4_dec
                cp    k_main
                jp    z,program_e
                cp    k_enter
                jp    z,program844
                call  key4_p4
                jr    program843
program844:     call  update
                ld    a,(s_g6)
                cp    1
                jp    z,up4
                call  no_update
                ld    a,(s_g6)
                cp    1
                jp    z,pro_f1
                ld    a,(b_mon)
                ld    (iy+6),a
```

```
                ld    a,(b_date)
                ld    (iy+7),a
                ld    a,(b_day)
                ld    (iy+8),a
                ld    a,(b_hour)
                ld    (iy+9),a
                ld    a,(b_min)
                ld    (iy+10),a
                ld    a,(b_sec)
                ld    (iy+11),a
                call  inc_ch
                inc   (ix+0)
                jp    pro_f1
p4_key1:        ld    a,(hl)
                and   0fh
                ld    (hl),a
                ld    a,c
                rlca
                rlca
                rlca
                rlca
                or    (hl)
                ld    (hl),a
                ret
p4_key2:        ld    a,(hl)
                and   0f0h
                ld    (hl),a
                ld    a,c
                or    (hl)
                ld    (hl),a
                ret
key_press4:     push  bc
                call  chang_key
                ld    a,(s_g5)
                cp    0
                jr    z,k4_pre_e
                ld    a,e
                cp    1
                call  z,p4_key1
                cp    2
                call  z,p4_key2
                ld    a,5
                call  gotoxy
                call  dis_f4
                pop   bc
                ld    a,c
                cp    10
                jr    z,k4_pre_e1
                inc   c
                ret
k4_pre_e:       pop   bc
k4_pre_e1:      ret
key4_p4:        push  bc
                call  chang_key
                ld    a,(s_g5)
                cp    0
                jr    z,k4_pre_e
                ld    a,e
                cp    1
                call  z,p4_key1
                cp    2
                call  z,p4_key2
                ld    a,69
                call  gotoxy
                call  dis_f4
                pop   bc
                ld    a,c
                cp    10
                jr    z,k4_pre_e1
                inc   c
                ret

dis_f4:         push  af
                push  bc
                push  de
                push  hl
                ld    hl,b_mon
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,"-"
                call  wrbyte
                ld    hl,b_date
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,20h
                call  wrbyte
                ld    hl,b_hour
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
                call  wrbyte
                ld    hl,b_min
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                ld    d,":"
                call  wrbyte
                ld    hl,b_sec
                ld    c,(hl)
                call  hex_ascii
                call  wrbyte
                call  hex_ascii
                ld    d,e
                call  wrbyte
                pop   hl
```

```
            pop   de
            pop   bc
            pop   af
            ret
p4_inc:     push  bc
            ld    a,e
            cp    1
            call  z,p4_inc1
            cp    2
            call  z,p4_inc2
            pop   bc
            ld    a,5
            call  gotoxy
            call  dis_f4
            ret
p4_dec:     push  bc
            ld    a,e
            cp    1
            call  z,p4_dec1
            cp    2
            call  z,p4_dec2
            pop   bc
            ld    a,5
            call  gotoxy
            call  dis_f4
            ret
op4_inc:    push  bc
            ld    a,e
            cp    1
            call  z,p4_inc1
            cp    2
            call  z,p4_inc2
            pop   bc
            ld    a,69
            call  gotoxy
            call  dis_f4
            ret
op4_dec:    push  bc
            ld    a,e
            cp    1
            call  z,p4_dec1
            cp    2
            call  z,p4_dec2
            pop   bc
            ld    a,69
            call  gotoxy
            call  dis_f4
            ret
p4_inc1:    ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    b
            call  z,p4_inc10
            inc   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p4_inc10:   ld    a,d
            dec   a
            ret
p4_inc2:    ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    b
            call  z,p4_inc10
            inc   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p4_dec1:    ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    d
            call  z,p4_dec10
            dec   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p4_dec10:   ld    a,b
            inc   a
            ret
p4_dec2:    ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    d
            call  z,p4_dec10
            dec   a
```

```
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
p4_left:    dec   c
            ld    a,c
            cp    0
            call  z,p4_left_d
            ret
p4_left_d:  ld    c,10
            ret
p4_right:   inc   c
            ld    a,c
            cp    11
            call  z,p4_left_t
            ret
p4_left_t:  ld    c,1
            ret
ota_p4:     push  ix
            ld    a,c
            cp    1
            call  z,ot4_tbl1
            cp    2
            call  z,ot4_tbl2
            cp    3
            call  z,ot4_tbl3
            cp    4
            call  z,ot4_tbl4
            cp    5
            call  z,ot4_tbl5
            cp    6
            call  z,ot4_tbl6
            cp    7
            call  z,ot4_tbl7
            cp    8
            call  z,ot4_tbl8
            cp    9
            call  z,ot4_tbl9
            cp    10
            call  z,ot4_tbla
            ld    a,(ix+0)
            call  gotoxy
            ld    d,(ix+1)
            ld    b,(ix+2)
            ld    e,(ix+3)
            pop   ix
            ret
ot4_tbl1:   ld    ix,otbl4_1
            ld    hl,b_mon
            ret
ot4_tbl2:   ld    ix,otbl4_2
            ld    hl,b_mon
            ret
ot4_tbl3:   ld    ix,otbl4_3
            ld    hl,b_date
            ret
ot4_tbl4:   ld    ix,otbl4_4
            ld    hl,b_date
            ret
ot4_tbl5:   ld    ix,otbl4_5
            ld    hl,b_hour
            ret
ot4_tbl6:   ld    ix,otbl4_6
            ld    hl,b_hour
            ret
ot4_tbl7:   ld    ix,otbl4_7
            ld    hl,b_min
            ret
ot4_tbl8:   ld    ix,otbl4_8
            ld    hl,b_min
            ret
ot4_tbl9:   ld    ix,otbl4_9
            ld    hl,b_sec
            ret
ot4_tbla:   ld    ix,otbl4_a
            ld    hl,b_sec
            ret
ta_p4:      push  ix
table_p4:   ld    a,c
            cp    1
            call  z,t4_tbl1
            cp    2
            call  z,t4_tbl2
            cp    3
            call  z,t4_tbl3
            cp    4
            call  z,t4_tbl4
            cp    5
            call  z,t4_tbl5
            cp    6
            call  z,t4_tbl6
            cp    7
            call  z,t4_tbl7
            cp    8
            call  z,t4_tbl8
            cp    9
            call  z,t4_tbl9
            cp    10
            call  z,t4_tbla
            ld    a,(ix+0)
            call  gotoxy
            ld    d,(ix+1)
            ld    b,(ix+2)
            ld    e,(ix+3)
            pop   ix
            ret
t4_tbl1:    ld    ix,tbl4_1
            ld    hl,b_mon
            ret
t4_tbl2:    ld    ix,tbl4_2
            ld    hl,b_mon
            ret
t4_tbl3:    ld    ix,tbl4_3
            ld    hl,b_date
            ret
t4_tbl4:    ld    ix,tbl4_4
            ld    hl,b_date
            ret
```

```
t4_tbl5:    ld    ix,tbl4_5
            ld    hl,b_hour
            ret
t4_tbl6:    ld    ix,tbl4_6
            ld    hl,b_hour
            ret
t4_tbl7:    ld    ix,tbl4_7
            ld    hl,b_min
            ret
t4_tbl8:    ld    ix,tbl4_8
            ld    hl,b_min
            ret
t4_tbl9:    ld    ix,tbl4_9
            ld    hl,b_sec
            ret
t4_tbla:    ld    ix,tbl4_a
            ld    hl,b_sec
            ret
tbl4_1:     dfb   5,0,1,1
tbl4_2:     dfb   6,0,9,2
tbl4_3:     dfb   8,0,3,1
tbl4_4:     dfb   9,0,9,2
tbl4_5:     dfb   11,0,2,1
tbl4_6:     dfb   12,0,9,2
tbl4_7:     dfb   14,0,5,1
tbl4_8:     dfb   15,0,9,2
tbl4_9:     dfb   17,0,5,1
tbl4_a:     dfb   18,0,9,2
otbl4_1:    dfb   69,0,1,1
otbl4_2:    dfb   70,0,9,2
otbl4_3:    dfb   72,0,3,1
otbl4_4:    dfb   73,0,9,2
otbl4_5:    dfb   75,0,2,1
otbl4_6:    dfb   76,0,9,2
otbl4_7:    dfb   78,0,5,1
otbl4_8:    dfb   79,0,9,2
otbl4_9:    dfb   81,0,5,1
otbl4_a:    dfb   82,0,9,2
program5:   ld    a,c
            cp    1
            call  z,program51
            dec   c
            ld    a,48h
            add   a,c
            call  gotoxy
            ret
program51:  ld    c,5   ;9
            ret
program6:   ld    a,c
            cp    4     ;8
            call  z,program61
            inc   c
            ld    a,48h
            add   a,c
            call  gotoxy
            ret
program61:  ld    c,0
            ret
program2:   ld    a,c
            cp    1
            call  z,program21
            dec   c
            ld    a,8
            add   a,c
            call  gotoxy
            ret
program21:  ld    c,9
            ret
program3:   ld    a,c
            cp    8
            call  z,program31
            inc   c
            ld    a,8
            add   a,c
            call  gotoxy
            ret
program31:  ld    c,0
            ret
prog_t:     ld    a,11
            call  gotoxy
program_e:  call  off_cursor
            call  delay
            call  clrscr
            call  delay
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
 inc_ch:    push  bc
            ld    b,12
 pro_ch1:   inc   iy
            djnz  pro_ch1
            pop   bc
            ret
dec_ch:     push  bc
            ld    b,12
pro_ch2:    dec   iy
            djnz  pro_ch2
            pop   bc
            ret
;*********************
;*  Set time to RTC  *
;*********************
set_time:   push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  delay
            call  clrscr
            call  delay
            ld    b,2
            ld    c,0
            ld    hl,st_time
            call  print
            call  delay
```

```
                call  dis_date
                call  dis_time
                call  delay
                call  on_cursor
                ld    c,1
set_time1:      call  time_set
                call  scan_key
                push  ix
                ld    ix,s_g3
                ld    (ix+0),a
                pop   ix
                cp    k_inc
                call  z,time_inc
                cp    k_dec
                call  z,time_dec
                cp    k_left
                call  z,time_left
                cp    k_right
                call  z,time_right
                cp    k_main
                jp    z,set_t_end1
                cp    k_enter
                jp    z,set_t_end
                ld    a,c
                cp    2
                call  z,set_tk
                ld    a,c
                cp    3
                call  z,set_tk
                ld    a,c
                cp    5
                call  z,set_tk
                ld    a,c
                cp    6
                call  z,set_tk
                ld    a,c
                cp    7
                call  z,set_tk
                ld    a,c
                cp    8
                call  z,set_tk
                ld    a,c
                cp    9
                call  z,set_tk
                ld    a,c
                cp    10
                call  z,set_tk
                jp    set_time1
set_t_end:      ld    a,(b_sec)
                out   (sec),a
                ld    a,(b_min)
                out   (minute),a
                ld    a,(b_hour)
                out   (hour),a
                ld    a,(b_day)
                out   (day),a
                ld    a,(b_date)
                out   (date),a
                ld    a,(b_mon)
                out   (month),a
set_t_end1:     call  off_cursor
                pop   iy
                pop   ix
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
set_tk:         push  bc
                push  ix
                ld    ix,s_g3
                ld    a,(ix+0)
                pop   ix
                ld    c,0
                cp    k_14
                jr    z,set0
                cp    k_1
                jr    z,set1
                cp    k_2
                jr    z,set2
                cp    k_3
                jr    z,set3
                cp    k_5
                jr    z,set4
                cp    k_6
                jr    z,set5
                cp    k_7
                jr    z,set6
                cp    k_9
                jr    z,set7
                cp    k_10
                jr    z,set8
                cp    k_11
                jr    z,set9
                push  ix
                ld    ix,s_g3
                ld    a,(ix+0)
                pop   ix
                pop   bc
                ret
set9:           inc   c
set8:           inc   c
set7:           inc   c
set6:           inc   c
set5:           inc   c
set4:           inc   c
set3:           inc   c
set2:           inc   c
set1:           inc   c
set0:           ld    a,e
                cp    1
                call  z,set10
                cp    2
                call  z,set11
                call  dis_date
                call  dis_time
                pop   bc
                push  ix
                ld    ix,s_g3
                ld    a,(ix+0)
```

```
            pop   ix
            ret
set12:      ld    c,0
            ret
set10:      ld    a,c
            rlca
            rlca
            rlca
            rlca
            ld    c,a
            ld    a,(hl)
            and   0fh
            or    c
            ld    (hl),a
            ret
set11:      ld    a,(hl)
            and   0f0h
            or    c
            ld    (hl),a
            ret
time_dec:   ld    a,c
            cp    4
            jp    z,time_dec4
            ld    a,e
            cp    1
            call  z,time_dec1
            cp    2
            call  z,time_dec2
            call  dis_date
            call  dis_time
            ret
time_dec1:  ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    d
            call  z,time_dec10
            dec   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
time_dec10: ld    a,b
            inc   a
            ret
time_dec2:  ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    d
            call  z,time_dec10
            dec   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
time_dec4:  ld    a,(hl)
            cp    10h
            jr    z,time_dec40
            cp    d
            jr    z,time_dec41
            dec   (hl)
            call  dis_date
            ret
time_dec40: ld    a,9
            ld    (hl),a
            call  dis_date
            ret
time_dec41: ld    a,b
            ld    (hl),a
            call  dis_date
            ret
time_inc:   ld    a,c
            cp    4
            jp    z,time_inc3
            ld    a,e
            cp    1
            call  z,time_inc1
            cp    2
            call  z,time_inc2
            call  dis_date
            call  dis_time
            ret
time_inc1:  ld    a,(hl)
            and   0fh
            ld    (b_set_time),a
            ld    a,(hl)
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            cp    b
            call  z,time_inc10
            inc   a
            rlca
            rlca
            rlca
            rlca
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
time_inc10: ld    a,d
```

```
            dec   a
            ret
time_inc2:  ld    a,(hl)
            and   0f0h
            ld    (b_set_time),a
            ld    a,(hl)
            and   0fh
            cp    b
            call  z,time_inc10
            inc   a
            push  hl
            ld    hl,b_set_time
            or    (hl)
            pop   hl
            ld    (hl),a
            ret
time_inc3:  ld    a,(hl)
            cp    9
            jr    z,time_inc30
            cp    b
            jr    z,time_inc31
            inc   (hl)
            call  dis_date
            ret
time_inc30: ld    a,10h
            ld    (hl),a
            call  dis_date
            ret
time_inc31: ld    a,d
            ld    (hl),a
            call  dis_date
            ret
time_left:  ld    a,c
            cp    1
            call  z,time_left1
            dec   c
            ret
time_left1: ld    c,11
            ret
time_right: ld    a,c
            cp    10
            call  z,time_rig
            inc   c
            ret
time_rig:   ld    c,0
            ret
time_set:   ld    a,c
            cp    1
            jr    z,time_set1
            cp    2
            jp    z,time_set2
            cp    3
            jr    z,time_set3
            cp    4
            jr    z,time_set4
            cp    5
            jr    z,time_set5
            cp    6
            jr    z,time_set6
            cp    7
            jr    z,time_set7
            cp    8
            jr    z,time_set8
            cp    9
            jr    z,time_set9
            cp    10
            jp    z,time_set10
            ret
time_set1:  ld    hl,b_day
            ld    d,1
            ld    b,7
            ld    e,2
            ld    a,0
            call  gotoxy
            ret
time_set2:  ld    hl,b_date
            ld    d,0
            ld    b,3
            ld    e,1
            ld    a,4
            call  gotoxy
            ret
time_set3:  ld    hl,b_date
            ld    d,0
            ld    b,9
            ld    e,2
            ld    a,5
            call  gotoxy
            ret
time_set4:  ld    hl,b_mon
            ld    d,1
            ld    b,12h
            ld    e,2
            ld    a,7
            call  gotoxy
            ret
time_set5:  ld    hl,b_hour
            ld    d,0
            ld    b,2
            ld    e,1
            ld    a,12
            call  gotoxy
            ret
time_set6:  ld    hl,b_hour
            ld    d,0
            ld    b,9
            ld    e,2
            ld    a,13
            call  gotoxy
            ret
time_set7:  ld    hl,b_min
            ld    d,0
            ld    b,5
            ld    e,1
            ld    a,15
            call  gotoxy
            ret
time_set8:  ld    hl,b_min
            ld    d,0
            ld    b,9
```

```
            ld    e,2
            ld    a,16
            call  gotoxy
            ret
time_set9:  ld    hl,b_sec
            ld    d,0
            ld    b,5
            ld    e,1
            ld    a,18
            call  gotoxy
            ret
time_set10: ld    hl,b_sec
            ld    d,0
            ld    b,9
            ld    e,2
            ld    a,19
            call  gotoxy
            ret
;**************
;*  Power on  *
;**************
power_on:   push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  clrscr
            call  delay
            call  off_cursor
            ld    b,1
            ld    c,0
            ld    hl,kmitl
            call  print
            call  wait
            call  wait
            ld    b,2
            ld    hl,fac
            call  print
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  clrscr
            call  delay
            ld    b,1
            ld    c,0
            ld    hl,pro_by
            call  print
            call  wait
            call  wait
            ld    b,2
            ld    hl,sang
            call  print
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  clrscr
            call  delay
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
;**************
;*  Test RAM  *
;**************
test_ram:   push  af
            push  bc
            push  de
            push  hl
            push  ix
            push  iy
            call  clrscr
            call  delay
            call  off_cursor
            ld    b,1
            ld    c,2
            ld    hl,st_ram
            call  print

            ld    hl,8000h

test_ram1:  ld    a,11
            call  gotoxy
            ld    c,h
            call  hex_ascii
            call  wrbyte
            ld    d,e
            call  wrbyte
            ld    c,l
            call  hex_ascii
            call  wrbyte
            ld    d,e
            call  wrbyte

            ld    a,(hl)
            push  af
            ld    a,0ffh
            ld    (hl),a
            ld    a,(hl)
            cp    0ffh
            jp    nz,ram_error
            ld    a,0
            ld    (hl),a
            ld    a,(hl)
            cp    0
            jp    nz,ram_error
```

```
            pop   af
            ld    (hl),a
            ld    de,10h
            add   hl,de
            ld    a,h
            cp    09fh
            jp    nz,test_ram1
            ld    b,2
            ld    c,2
            ld    hl,st_ram1
            call  print
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  wait
            call  clrscr
            call  delay
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
ram_error:  pop   af
            ld    b,2
            ld    c,0
            ld    hl,st_ram2
            call  print
            halt
;*******************
;*  BINARY TO BCD  *
;*  INPUT   =  B   *
;*  OUT     =  B   *
;*******************
bin_bcd:    push  af
            ld    a,b
            cp    0
            jr    z,bin_bcd_e
            ld    a,0
aaaa1:      inc   a
            daa
            djnz  aaaa1
            ld    b,a
bin_bcd_e:  pop   af
            ret
;****************
;*  Wait state  *
;****************
wait:       push  bc
            ld    b,0ffh
wait1:      ld    c,0ffh
wait2:      dec   c
            jr    nz,wait2
            djnz  wait1
            pop   af
            ret
;*************************
;*  HEX to ASCII convert  *
;*  IN  = C  HEX input    *
;*  OUT = DE  ascii       *
;*************************
hex_ascii:  push  af
            push  bc
            push  hl
            ld    a,c
            and   0f0h
            rrca
            rrca
            rrca
            rrca
            call  hex_ascii1
            ld    d,a
            ld    a,c
            call  hex_ascii1
            ld    e,a
            pop   hl
            pop   bc
            pop   af
            ret
hex_ascii1: and   0fh
            cp    0ah
            jr    nc,hex_ascii2
            or    30h
            ret
hex_ascii2: sub   9
            or    40h
            ret
            ;************************************
            ;*Change Key from scan to Number code*
            ;*  Input  = call scan_key          *
            ;*  Output = C                      *
            ;************************************
chang_key:  push  af
            push  de
            push  hl
            push  af
            ld    a,0
            ld    (s_g5),a
            pop   af
            ld    c,0
            cp    k_14
            jr    z,c0
            cp    k_1
            jr    z,c1
            cp    k_2
            jr    z,c2
            cp    k_3
            jr    z,c3
            cp    k_5
            jr    z,c4
            cp    k_6
            jr    z,c5
            cp    k_7
            jr    z,c6
            cp    k_9
            jr    z,c7
            cp    k_10
```

```
            jr    z,c8
            cp    k_11
            jr    z,c9
chang_end:  pop   hl
            pop   de
            pop   af
            ret
c9:         inc   c
c8:         inc   c
c7:         inc   c
c6:         inc   c
c5:         inc   c
c4:         inc   c
c3:         inc   c
c2:         inc   c
c1:         inc   c
c0:         ld    a,1
            ld    (s_g5),a
            jr    chang_end
;*********************************
;*Display data for LCD HDM- 20216H*
;*********************************
INITLCD:    LD    A,0
            OUT   (P_lcd2),A
            LD    A,00111000B
            OUT   (P_lcd1),A
            CALL  EPLUSE
            CALL  DELAY
            LD    A,00001111B
            OUT   (P_lcd1),A
            CALL  EPLUSE
            LD    A,00000110B
            OUT   (P_lcd1),A
            CALL  EPLUSE
            LD    A,00000001B
            OUT   (P_lcd1),A
            CALL  EPLUSE
            CALL  DELAY
            RET
;****************
;*  OFF cursor  *
;****************
off_cursor: push  af
            ld    a,0
            out   (P_lcd2),a
            ld    a,00001100b
            out   (P_lcd1),a
            call  epluse
            pop   af
            ret
;***************
;*  ON cursor  *
;***************
on_cursor:  push  af
            ld    a,0
            out   (P_lcd2),a
            ld    a,00001111b
            out   (P_lcd1),a
            call  epluse
            pop   af
            ret
;****************
;*  Back space  *
;****************
back_space: push  af
            ld    a,0
            out   (P_lcd2),a
            ld    a,00010000b
            out   (P_lcd1),a
            call  epluse
            pop   af
            ret
;******************
;*  Clear screen  *
;******************
clrscr:     push  af
            ld    a,0
            out   (P_lcd2),a
            ld    a,00000001b
            out   (P_lcd1),a
            call  epluse
            pop   af
            ret
;*********************************
;*  Write line lcd 20 charecter  *
;*  IN  = HL  data display       *
;*        A   line display       *
;*********************************
WRLINE:     CP    1
            JR    Z,WRL1
            CP    2
            JR    Z,WRL2
            RET
WRL1:       LD    a,0
            call  gotoxy
            JR    WRLM
WRL2:       LD    A,40H
            call  gotoxy
WRLM:       LD    B,20
WRL:        LD    D,(HL)
            PUSH  BC
            CALL  WRBYTE
            POP   BC
            INC   HL
            DJNZ  WRL
            RET
;****************************
;*  Enable pluse generator  *
;****************************
epluse:     push  af
            push  bc
            in    a,(P_lcd2)
            sett  2,a
            out   (P_lcd2),a
            ld    b,0
epl:        djnz  epl
            res   2,a
            out   (P_lcd2),a
            pop   bc
            pop   af
```

```
            ret
;*************************
;*  Gotoxy               *
;*  IN  = A              *
;*  Line 1 = 00h-13h     *
;*  Line 2 = 40h-53h     *
;*************************
gotoxy:     PUSH   BC
            SETT   7,A
            OUT    (P_lcd1),A
            XOR    A
            OUT    (P_lcd2),A
            CALL   EPLUSE
            POP    BC
            RET
;*******************************
;*  write data to lcd display  *
;*  IN  = D  'charecter'       *
;*  OUT = display              *
;*******************************
wrbyte:     push   af
            ld     a,00000001B
            out    (P_lcd2),a
            ld     a,d
            out    (P_lcd1),a
            call   epluse
            pop    af
            ret
;************************
;*  Delay data for LCD  *
;************************
DELAY:      LD     B,0
DE1:        NOP
            NOP
            DJNZ   DE1
            RET
;*******************************
;*  Move cursor top and left   *
;*******************************
home:       push   af
            push   bc
            ld     a,80h
            out    (P_lcd1),a
            ld     a,0
            out    (P_lcd2),a
            call   epluse
            pop    bc
            pop    af
            ret
;************************
;*  Print "data"        *
;*  IN  = B  'line'     *
;*        C  'colum'    *
;*        HL 'data'     *
;*  OUT = display       *
;************************
Print:      push   af
            push   de
            ld     a,b
            cp     1
            jr     z,print_1
            ld     a,40h
            add    a,c
            call   gotoxy
            jp     print_2
print_1:    ld     a,0
            add    a,c
            call   gotoxy
print_2:    ld     a,(hl)
            cp     24h
            jp     z,print_end
            ld     d,a
            push   bc
            call   wrbyte
            pop    bc
            inc    hl
            jp     print_2
print_end:  pop    de
            pop    af
            ret
;************************
;*  Load time form RTC  *
;*     to buffer        *
;************************
time:       push   af
            in     a,(sec10)
            ld     (b_sec10),a
            in     a,(sec)
            ld     (b_sec),a
            in     a,(minute)
            ld     (b_min),a
            in     a,(hour)
            ld     (b_hour),a
            in     a,(day)
            ld     (b_day),a
            in     a,(date)
            ld     (b_date),a
            in     a,(month)
            ld     (b_mon),a
            pop    af
            ret
;*****************
;* display time  *
;*****************
dis_time:   push   af
            push   bc
            push   de
            push   hl
            push   ix
            push   iy
            ld     a,":"
            ld     hl,disb_time+2
            ld     (hl),a
            ld     hl,disb_time+5
            ld     (hl),a
            ld     hl,disb_time+8
            ld     a,"$"
            ld     (hl),a
            ld     hl,disb_time
            ld     ix,b_hour
```

```
                ld    b,3
d_time10:       ld    a,(ix+0)
                ld    e,a
                and   0f0h
                rrca
                rrca
                rrca
                rrca
                add   a,30h
                ld    (hl),a
                inc   hl
                ld    a,e
                and   0fh
                add   a,30h
                ld    (hl),a
                inc   hl
                inc   hl
                dec   ix
                djnz  d_time10

                ld    b,1
                ld    c,12
                ld    hl,disb_time
                call  print
dis_t_end:      pop   iy
                pop   ix
                pop   hl
                pop   de
                pop   bc
                pop   af
                ret
;******************
;*  Display date  *
;******************
dis_date:       push  af
                push  bc
                push  de
                push  hl
                push  ix
                push  iy
                ld    de,4
                ld    hl,sun
                ld    a,(b_day)
                cp    1
                jp    z,dis_date1
                cp    2
                jp    z,dis_date2
                cp    3
                jp    z,dis_date3
                cp    4
                jp    z,dis_date4
                cp    5
                jp    z,dis_date5
                cp    6
                jp    z,dis_date6
                cp    7
                jp    z,dis_date7
dis_date7:      add   hl,de
dis_date6:      add   hl,de
dis_date5:      add   hl,de
dis_date4:      add   hl,de
dis_date3:      add   hl,de
dis_date2:      add   hl,de
dis_date1:      ld    b,1
                ld    c,0
                call  print
                ld    d,"-"
                call  wrbyte
                ld    a,(b_date)
                ld    e,a
                and   0f0h
                rrca
                rrca
                rrca
                rrca
                or    30h
                ld    d,a
                call  wrbyte
                ld    a,e
                and   0fh
                or    30h
                ld    d,a
                call  wrbyte
                ld    d,"-"
                call  wrbyte
                ld    de,4
                ld    hl,jan
                ld    a,(b_mon)
                cp    1
                jp    z,dis_date11
                cp    2
                jp    z,dis_date12
                cp    3
                jp    z,dis_date13
                cp    4
                jp    z,dis_date14
                cp    5
                jp    z,dis_date15
                cp    6
                jp    z,dis_date16
                cp    7
                jp    z,dis_date17
                cp    8
                jp    z,dis_date18
                cp    9
                jp    z,dis_date19
                cp    10h
                jp    z,dis_date20
                cp    11h
                jp    z,dis_date21
                cp    12h
                jp    z,dis_date22
dis_date22:     add   hl,de
dis_date21:     add   hl,de
dis_date20:     add   hl,de
dis_date19:     add   hl,de
dis_date18:     add   hl,de
dis_date17:     add   hl,de
dis_date16:     add   hl,de
dis_date15:     add   hl,de
```

```
dis_date14: add   hl,de
dis_date13: add   hl,de
dis_date12: add   hl,de
dis_date11: ld    b,1
            ld    c,7
            call  print
            pop   iy
            pop   ix
            pop   hl
            pop   de
            pop   bc
            pop   af
            ret
;********************
;*   In key board   *
;* IN  = -          *
;* OUT = A          *
;********************
scan_key:     push  bc
              push  de
              push  hl
scan_key0:    ld    b,3           ;ld b=3 for 3 row
              ld    d,0feh        ;set bit 0 select
scan_key1:    ld    a,d
              out   (P_key),a     ;out to control
              rlc   d
              in    a,(P_key)     ;in data key
              ld    e,a
              and   0f0h
              cp    0f0h
              jp    nz,scan_key2  ;if key push bottom
              djnz  scan_key1     ;if no key jump to rotate row
              ld    a,0ffh        ;if no key result 0FFH to buff
              ld    (key),a
              jp    scan_key3
scan_key2:    ld    a,e
              ld    hl,key
              cp    (hl)          ;compear old key for make bottom
              jp    z,scan_key4
              ld    (key),a
scan_key3:    pop   hl
              pop   de
              pop   bc
              ret
scan_key4:    ld    a,0ffh        ;if old key
              jp    scan_key3
;***************************
;* Buffer data for display *
;***************************
sun:        dfb   "SUN$"
mon:        dfb   "MON$"
tue:        dfb   "TUE$"
wed:        dfb   "WED$"
thu:        dfb   "THU$"
fri:        dfb   "Fri$"
sat:        dfb   "SAT$"
jan:        dfb   "JAN$"
feb:        dfb   "FAB$"
mar:        dfb   "MAR$"
apr:        dfb   "APR$"
```

```
may:        dfb   "MAY$"
jun:        dfb   "JUN$"
jul:        dfb   "JUL$"
aug:        dfb   "AUG$"
sep:        dfb   "SEP$"
oct:        dfb   "OCT$"
nov:        dfb   "NOV$"
dec:        dfb   "Dec$"
st_ram:     dfb   "TEST RAM$"
st_ram1:    dfb   "..TEST RAM OK..$"
st_ram2:    dfb   "Ram ERROR change RAM$"
kmitl:      dfb   "  KMIT LADKRABANG$"
fac:        dfb   "Faculty of Engineer "
pro_by:     dfb   "     Program by$"
sang:       dfb   "  Sangkom & Somkid$"
s1_begin:   dfb   "   Clear all data.$"
s2_begin:   dfb   "   Don't test RAM.$"
s3_begin:   dfb   "  Don't clear data.$"
st_time:    dfb   "   Enter new time $"
s1_pc1:     dfb   "  Load data from PC$"
s2_pc1:     dfb   "   Enter to start$"
s3_pc1:     dfb   "   Ready   OK..!$"
s4_pc1:     dfb   "  Start code ERROR$"
s5_pc1:     dfb   "  Start code OK..$"
s6_pc1:     dfb   "   End of loading$"
sman_1:     dfb   "       Manual$"
sman_2:     dfb   "   On  Off  Cancel$"
s_no:       dfb   "   Data your BAD !!$"
s_no1:      dfb   "                    $"
sup_1:      dfb   "Month  ERROR ! $"
sup_2:      dfb   "Date   ERROR ! $"
sup_3:      dfb   "Day    ERROR ! $"
sup_4:      dfb   "Hour   ERROR ! $"
sup_5:      dfb   "Minute ERROR ! $"
sup_6:      dfb   "Second ERROR ! $"
sup_7:      dfb   "                    $"
st_c1:      dfb   "   Clear PROGRAM$"
st_c2:      dfb   "  Clear channel $"
st_c3:      dfb   "      OK..$"
st_p1:      dfb   " Channel 12345678$"
st_p2:      dfb   "Function 1234$"
st_p3:      dfb   " Total     PROGRAM$"
st_p4:      dfb   "  Press enter key$"
st_pf1:     dfb   "ON   $" ;    00:00:00 $"
st_pf11:    dfb   "OFF  $" ;    00:00:00 $"

            END
```

Zilog

Z-80® PIO
Z-80A PIO

Product Specification

The Zilog Z-80 product line is a complete set of microcomputer components, development systems and support software. The Z-80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

The Z-80 Parallel I/O (PIO) Interface Controller is a programmable, two port device which provides TTL compatible interfacing between peripheral devices and the Z80-CPU. The Z80-CPU configures the Z80-PIO to interface with standard peripheral devices such as tape punches, printers, keyboards, etc.

Structure

- N-Channel Silicon Gate Depletion Load technology
- 40 Pin DIP
- Single 5 volt supply
- Single phase 5 volt clock
- Two independent 8-bit bidirectional peripheral interface ports with "handshake" data transfer control

catures

- Interrupt driven "handshake" for fast response
- Any one of the following modes of operation may be selected for either port:
  - Byte output
  - Byte input
  - Byte bidirectional bus (available on Port A only)
  - Bit Mode
- Programmable interrupts on peripheral status conditions.
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.
- Eight outputs are capable of driving Darlington transistors.
- All inputs and outputs fully TTL compatible.

PIO Architecture

A block diagram of the Z80-PIO is shown in figure 1. The internal structure of the Z80-PIO consists of a Z80-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. A typical application might use Port A as the data transfer channel and Port B for the status and control monitoring.

The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in figure 2. The registers include: an 8-bit input register, an 8-bit output register, a 2-bit mode control register, an 8-bit mask register, an 8-bit input/output select register, and a 2-bit mask control register. The last three registers are used only when the port has been programmed to operate in the bit mode.

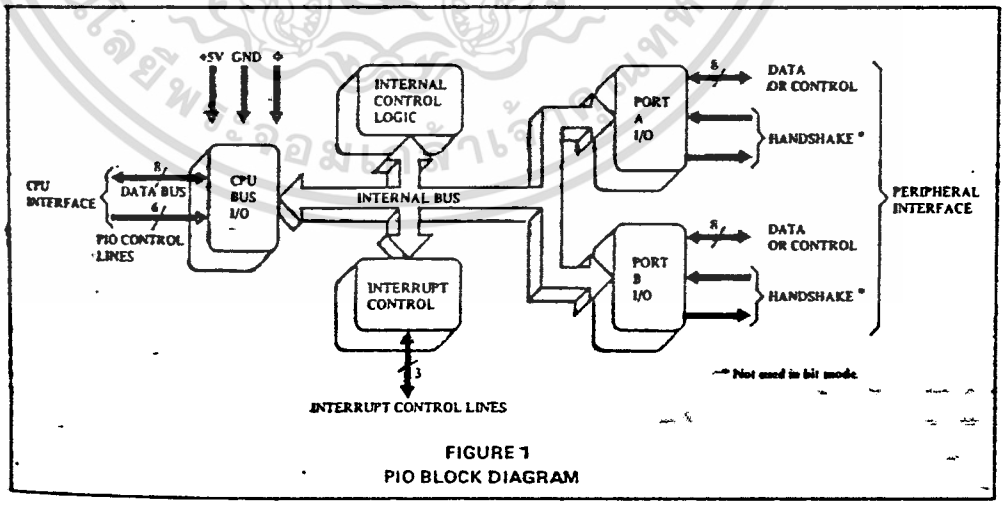


FIGURE 1
PIO BLOCK DIAGRAM

รายละเอียดข้อมูลไมโครโปรเซสเซอร์และชิพสนับสนุน

Z80®-CPU
Z80A-CPU

Product Specification

MARCH 1978

The Zilog Z80 product line is a complete set of microcomputer components, development systems and support software. The Z80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

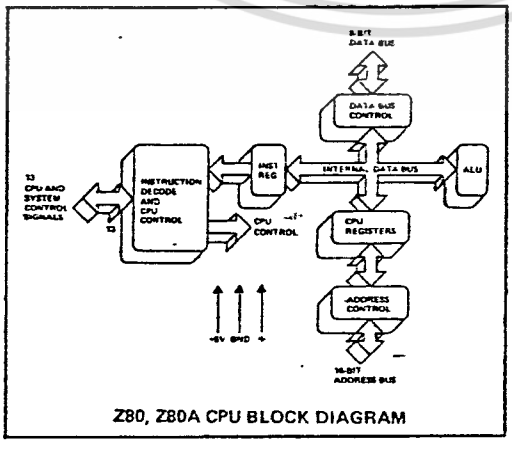
The Z80 and Z80A CPU's are third generation single chip microprocessors with unrivaled computational power. This increased computational power results in higher system through-put and more efficient memory utilization when compared to second generation microprocessors. In addition, the Z80 and Z80A CPU's are very easy to implement into a system because of their single voltage requirement plus all output signals are fully decoded and timed to control standard memory or peripheral circuits. The circuit is implemented using an N-channel, ion implanted, silicon gate MOS process.

Figure 1 is a block diagram of the CPU. Figure 2 details the internal register configuration which contains 208 bits of Read/Write memory that are accessible to the programmer. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or as 16-bit register pairs. There are also two sets of accumulator and flag registers. The programmer has access to either set of main or alternate registers through a group of exchange instructions. This alternate set allows foreground/background mode of operation or may be reserved for very fast Interrupt response. Each CPU also contains a 16-bit stack pointer which permits simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data handling.

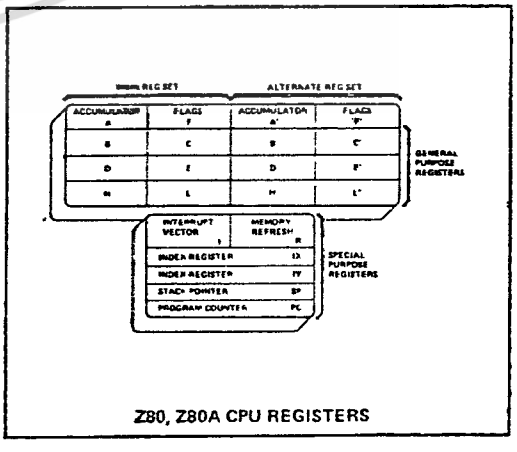
The two 16-bit index registers allow tabular data manipulation and easy implementation of relocatable code. The Refresh register provides for automatic, totally transparent refresh of external dynamic memories. The I register is used in a powerful interrupt response mode to form the upper 8 bits of a pointer to a interrupt service address table, while the interrupting device supplies the lower 8 bits of the pointer. An indirect call is then made to this service address.

FEATURES

- Single chip, N-channel Silicon Gate CPU.
- 158 instructions—includes all 78 of the 8080A instructions with total software compatibility. New instructions include 4-, 8- and 16-bit operations with more useful addressing modes such as indexed, bit and relative.
- 17 internal registers.
- Three modes of fast interrupt response plus a non-maskable interrupt.
- Directly interfaces standard speed static or dynamic memories with virtually no external logic.
- 1.0 μs instruction execution speed.
- Single 5 VDC supply and single-phase 5 volt Clock.
- Out-performs any other single chip microcomputer in 4-, 8-, or 16-bit applications.
- All pins TTL Compatible
- Built-in dynamic RAM refresh circuitry.

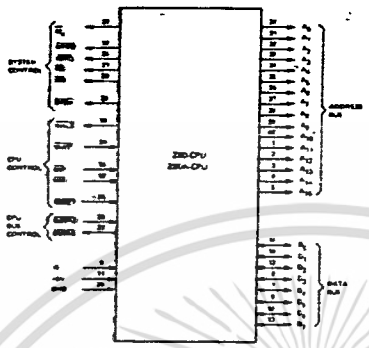


Z80, Z80A CPU BLOCK DIAGRAM



Z80, Z80A CPU REGISTERS

Z80, Z80A-CPU Pin Description



Z80, Z80A CPU PIN CONFIGURATION

A_0-A_{15} (Address Bus) Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges.

D_0-D_7 (Data Bus) Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one) Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution.

$\overline{MREQ}$ (Memory Request) Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{IORQ}$ (Input/ Output Request) Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{IORQ}$ signal is also generated when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus.

$\overline{RD}$ (Memory Read) Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{WR}$ (Memory Write) Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{RFSH}$ (Refresh) Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{HALT}$ (Halt state) Output, active low. $\overline{HALT}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{WAIT}$ (Wait) Input, active low. $\overline{WAIT}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active.

$\overline{INT}$ (Interrupt Request) Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled.

$\overline{NMI}$ (Non Maskable Interrupt) Input, active low. The non-maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z-80 CPU to restart to location 0066_H.

$\overline{RESET}$ Input, active low. $\overline{RESET}$ initializes the CPU as follows: reset interrupt enable flip-flop, clear PC and registers I and R and set interrupt to 8080A mode. During reset time, the address and data bus go to a high impedance state and all control output signals go to the inactive state.

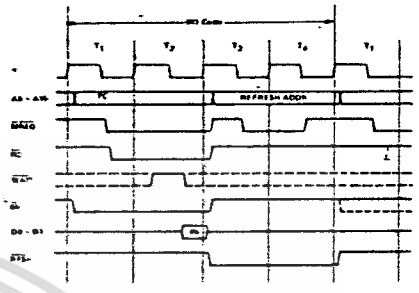
$\overline{BUSRQ}$ (Bus Request) Input, active low. The bus request signal has a higher priority than $\overline{NMI}$ and is always recognized at the end of the current machine cycle and is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these busses.

$\overline{BUSAK}$ (Bus Acknowledge) Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Timing Waveforms

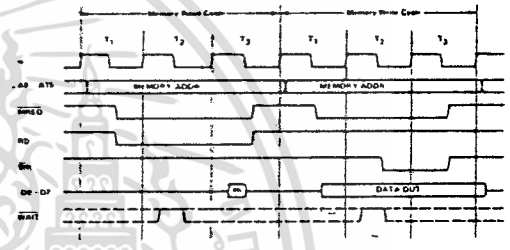
INSTRUCTION OP CODE FETCH

The program counter content (PC) is placed on the address bus immediately at the start of the cycle. One half clock time later $\overline{MREQ}$ goes active. The falling edge of $\overline{MREQ}$ can be used directly as a chip enable to dynamic memories. $\overline{RD}$ when active indicates that the memory data should be enabled onto the CPU data bus. The CPU samples data with the rising edge of the clock state T_3. Clock states T_3 and T_4 of a fetch cycle are used to refresh dynamic memories while the CPU is internally decoding and executing the instruction. The refresh control signal $\overline{RFSH}$ indicates that a refresh read of all dynamic memories should be accomplished.



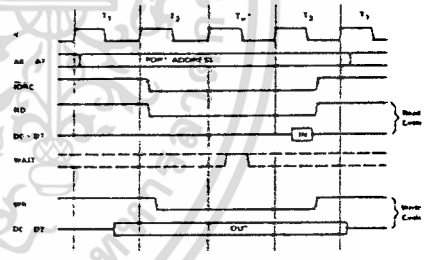
MEMORY READ OR WRITE CYCLES

Illustrated here is the timing of memory read or write cycles other than an OP code fetch (M_1 cycle). The $\overline{MREQ}$ and $\overline{RD}$ signals are used exactly as in the fetch cycle. In the case of a memory write cycle, the $\overline{MREQ}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{WR}$ line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory.



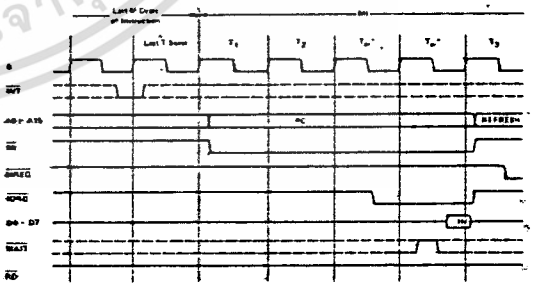
INPUT OR OUTPUT CYCLES

Illustrated here is the timing for an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted (Tw*). The reason for this is that during I/O operations this extra state allows sufficient time for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required.



INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

The interrupt signal is sampled by the CPU with the rising edge of the last clock at the end of any instruction. When an interrupt is accepted, a special M_1 cycle is generated. During this M_1 cycle, the $\overline{IORQ}$ signal becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Two wait states (Tw*) are automatically added to this cycle so that a ripple priority interrupt scheme, such as the one used in the Z80 peripheral controllers, can be easily implemented.



Z80, Z80A Instruction Set

The following is a summary of the Z80, Z80A instruction set showing the assembly language mnemonic and the symbolic operation performed by the instruction. A more detailed listing appears in the Z80-CPU technical manual, and assembly language programming manual. The instructions are divided into the following categories:

8-bit loads	Miscellaneous Group
16-bit loads	Rotates and Shifts
Exchanges	Bit Set, Reset and Test
Memory Block Moves	Input and Output
Memory Block Searches	Jumps
8-bit arithmetic and logic	Calls
16-bit arithmetic	Restarts
General purpose Accumulator & Flag Operations	Returns

In the table the following terminology is used.

b ≡ a bit number in any 8-bit register or memory location

cc ≡ flag condition code
- NZ ≡ non zero
- Z ≡ zero
- NC ≡ non carry
- C ≡ carry
- PO ≡ Parity odd or no over flow
- PE ≡ Parity even or over flow
- P ≡ Positive
- M ≡ Negative (minus)

d ≡ any 8-bit destination register or memory location

dd ≡ any 16-bit destination register or memory location

e ≡ 8-bit signed 2's complement displacement used in relative jumps and indexed addressing

L ≡ 8 special call locations in page zero. In decimal notation these are 0, 8, 16, 24, 32, 40, 48 and 56

n ≡ any 8-bit binary number

nn ≡ any 16-bit binary number

r ≡ any 8-bit general purpose register (A, B, C, D, E, H, or L)

s ≡ any 8-bit source register or memory location

s_b ≡ a bit in a specific 8-bit register or memory location

ss ≡ any 16-bit source register or memory location

subscript "L" ≡ the low order 8 bits of a 16-bit register

subscript "H" ≡ the high order 8 bits of a 16-bit register

( ) ≡ the contents within the ( ) are to be used as a pointer to a memory location or I/O port number

8-bit registers are A, B, C, D, E, H, L, I and R

16-bit register pairs are AF, BC, DE and HL

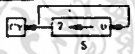
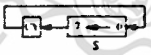
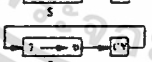
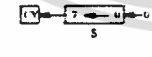
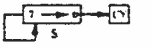
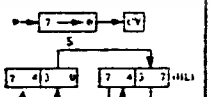
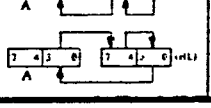
16-bit registers are SP, PC, IX and IY

Addressing Modes implemented include combinations of the following:

Immediate	Indexed
Immediate extended	Register
Modified Page Zero	Implied
Relative	Register Indirect
Extended	Bit

	Mnemonic	Symbolic Operation	Comments
8-BIT LOADS	LD r, s	r ← s	s ≡ r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d ≡ (HL), r, (IX+e), (IY+e)
	LD d, n	d ← n	d ≡ (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s ≡ (BC), (DE), (nn), I, R
	LD d, A	d ← A	d ≡ (BC), (DE), (nn), I, R
16-BIT LOADS	LD dd, nn	dd ← nn	dd ≡ BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd ≡ BC, DE, HL, SP, IX, IY
	LD (nn), ss	(nn) ← ss	ss ≡ BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss = HL, IX, IY
	PUSH ss	$(SP-1) \leftarrow ss_H$; $(SP-2) \leftarrow ss_L$	ss = BC, DE, HL, AF, IX, IY
	POP dd	$dd_L \leftarrow (SP)$; $dd_H \leftarrow (SP+1)$	dd = BC, DE, HL, AF, IX, IY
EXCHANGES	EX DE, HL	DE ↔ HL	
	EX AF, AF'	AF ↔ AF'	
	EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix} \leftrightarrow \begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	
	EX (SP), ss	$(SP) \leftarrow ss_L$, $(SP+1) \leftarrow ss_H$	ss ≡ HL, IX, IY

	Mnemonic	Symbolic Operation	Comments
MEMORY BLOCK MOVES	LDI	(DE) ← (HL), DE ← DE+1, HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1, HL ← HL+1, BC ← BC-1, Repeat until BC = 0	
	LDD	(DE) ← (HL), DE ← DE-1, HL ← HL-1, BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE-1, HL ← HL-1, BC ← BC-1, Repeat until BC = 0	
MEMORY BLOCK SEARCHES	CPI	A-(HL), HL ← HL+1, BC ← BC-1	A-(HL) sets the flags only. A is not affected
	CPIR	A-(HL), HL ← HL+1, BC ← BC-1, Repeat until BC = 0 or A = (HL)	
	CPD	A-(HL), HL ← HL-1, BC ← BC-1	
	CPDR	A-(HL), HL ← HL-1, BC ← BC-1, Repeat until BC = 0 or A = (HL)	
8-BIT ALU	ADD s	A ← A + s	CY is the carry flag s ≡ r, n, (HL), (IX+e), (IY+e)
	ADC s	A ← A + s + CY	
	SUB s	A ← A - s	
	SBC s	A ← A - s - CY	
	AND s	A ← A ∧ s	
	OR s	A ← A ∨ s	
	XOR s	A ← A ⊕ s	

	Mnemonic	Symbolic Operation	Comments
8-BIT ALU	CP s	A - s	s = r, n (HL) (IX+e), (IY+e)
	INC d	d ← d + 1	d = r, (HL) (IX+e), (IY+e)
	DEC d	d ← d - 1	
16-BIT ARITHMETIC	ADD HL, ss	HL ← HL + ss	ss ≡ BC, DE HL, SP
	ADC HL, ss	HL ← HL + ss + CY	
	SBC HL, ss	HL ← HL - ss - CY	
	ADD IX, ss	IX ← IX + ss	ss ≡ BC, DE, IX, SP
	ADD IY, ss	IY ← IY + ss	ss ≡ BC, DE, IY, SP
	INC dd	dd ← dd + 1	dd ≡ BC, DE, HL, SP, IX, IY
	DEC dd	dd ← dd - 1	dd ≡ BC, DE, HL, SP, IX, IY
GP ACC. & FLAG	DAA	Converts A contents into packed BCD following add or subtract	Operands must be in packed BCD format
	CPL	$A \leftarrow \overline{A}$	
	NEG	A ← 00 - A	
	CCF	$CY \leftarrow \overline{CY}$	
	SCF	CY ← 1	
MISCELLANEOUS	NOP	No operation	
	HALT	Halt CPU	
	DI	Disable Interrupts	
	EI	Enable Interrupts	
	IM 0	Set interrupt mode 0	8080A mode
	IM 1	Set interrupt mode 1	Call to 0038_H
	IM 2	Set interrupt mode 2	Indirect Call
ROTATES AND SHIFTS	RLC s		s ≡ r, (HL) (IX+e), (IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		

	Mnemonic	Symbolic Operation	Comments
BIT S, R, & T	BIT b, s	$Z \leftarrow \overline{s_b}$	Z is zero flag
	SET b, s	$s_b \leftarrow 1$	s ≡ r, (HL) (IX+e), (IY+e)
	RES b, s	$s_b \leftarrow 0$	
INPUT AND OUTPUT	IN A, (n)	A ← (n)	
	IN r, (C)	r ← (C)	Set flags
	INI	(HL) ← (C), HL ← HL + 1 B ← B - 1	
	INIR	(HL) ← (C), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	IND	(HL) ← (C), HL ← HL - 1 B ← B - 1	
	INDR	(HL) ← (C), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
	OUT(n), A	(n) ← A	
	OUT(C), r	(C) ← r	
	OUTI	(C) ← (HL), HL ← HL + 1 B ← B - 1	
	OTIR	(C) ← (HL), HL ← HL + 1 B ← B - 1 Repeat until B = 0	
	OUTD	(C) ← (HL), HL ← HL - 1 B ← B - 1	
	OTDR	(C) ← (HL), HL ← HL - 1 B ← B - 1 Repeat until B = 0	
JUMPS	JP nn	PC ← nn	cc ≡ NZ PO, Z PE, NC P, C M
	JP cc, nn	If condition cc is true PC ← nn, else continue	
	JR e	PC ← PC + e	kk ≡ NZ NC, Z C
	JR kk, e	If condition kk is true PC ← PC + e, else continue	
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B - 1, if B = 0 continue, else PC ← PC + e	
CALLS	CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, PC ← nn	cc ≡ NZ PO, Z PE, NC P, C M
	CALL cc, nn	If condition cc is false continue, else same as CALL nn	
RESTARTS	RST L	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$ $PC_L \leftarrow L$	
RETURNS	RET	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$	cc ≡ NZ PO, Z PE, NC P, C M
	RET cc	If condition cc is false continue, else same as RET	
	RETI	Return from interrupt, same as RET	
	RETN	Return from non-maskable interrupt	

A.C. Characteristics Z80-CPU

T_A = 0°C to 70°C, V_{CC} = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	4	1121	μsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180	[E]	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	t_r, t_f	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$, or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock $\overline{MREQ}$ Low		100	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_w(MRL)$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_w(MRH)$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		100	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		100	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		110	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		90	nsec	
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_w(\overline{WR}L)$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		130	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		130	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		180	nsec	C_L = 50pF
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		150	nsec	
$\overline{WAIT}$	$t_{S(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{INT}$	$t_{S(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_w(\overline{NML})$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{S(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		120	nsec	C_L = 50pF
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		110	nsec	
$\overline{RESET}$	$t_{S(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	
	t_{ssi}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 210$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$

[8] $t_w(MRL) = t_c - 40$

[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(\overline{WR}L)} = t_c - 40$

[11] $t_{ssi} = 2t_c + t_{w(\Phi H)} + t_f - 80$

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

NOTES

A Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B All control signals are internally synchronized so they may be totally asynchronous with respect to the clock.

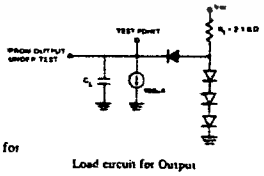
C The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

D Output Delay vs. Loaded Capacitance

TA = 70°C V_{CC} = +5V ±5%

Add 10nsec delay for each 50pf increase in load up to a maximum of 200pf for the data bus & 100pf for address & control lines

E Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum.

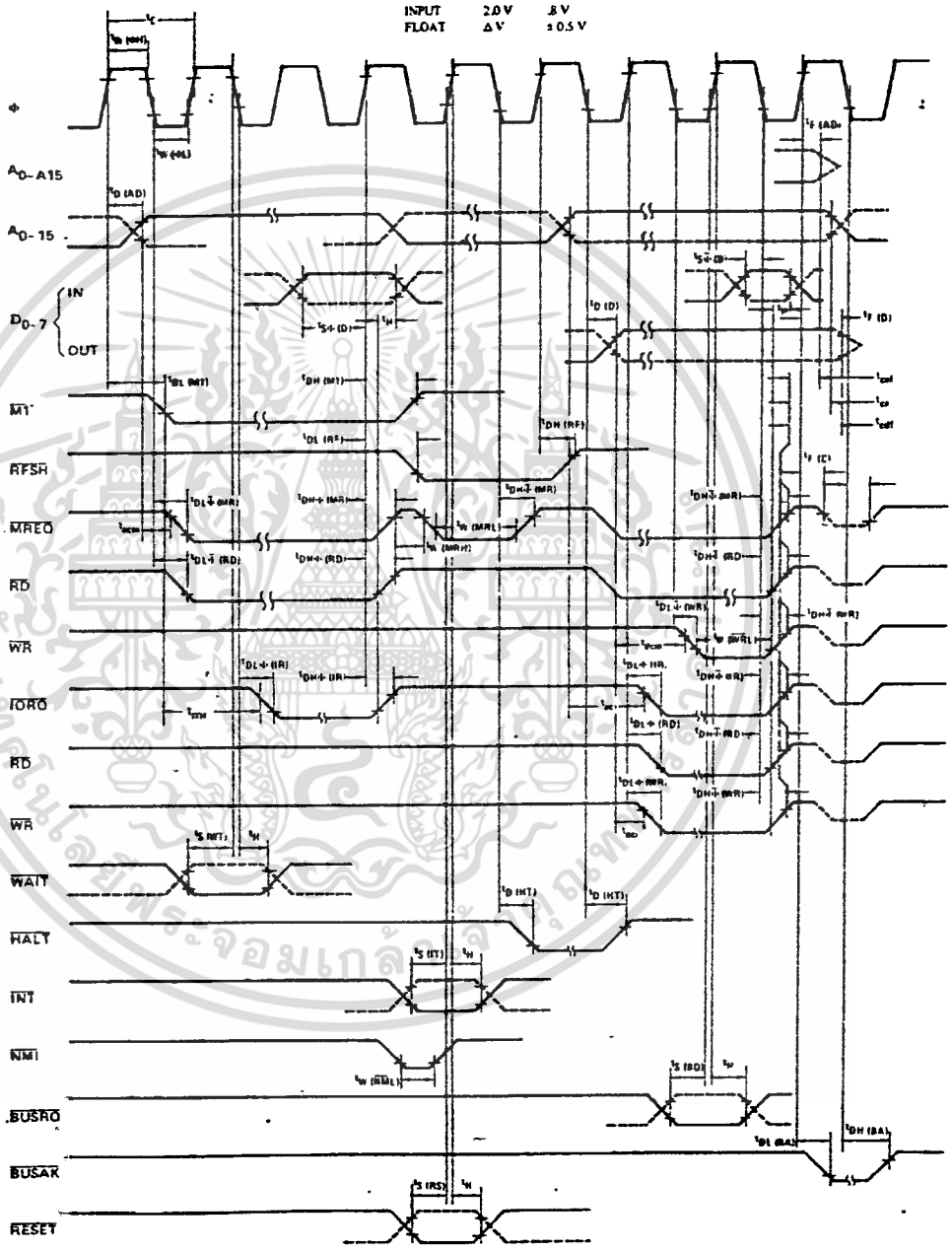


Load circuit for Output

A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{CC} -.6V	.45V
OUTPUT	2.0 V	.8 V
INPUT	2.0 V	.8 V
FLOAT	Δ V	± 0.5 V



Absolute Maximum Ratings

Temperature Under Bias	Specified operating range.
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note For Z80-CPU all AC and DC characteristics remain the same for the military grade parts except I_{CC}

I_{CC} = 200 mA

Z80-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} -.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH}=-250μA
I_{CC}	Power Supply Current			150	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \le V_{IN} \le V_{CC}$

Z80A-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} -.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH}=-250μA
I_{CC}	Power Supply Current		90	200	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \le V_{IN} \le V_{CC}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C
E – Extended 5V ±5% -40° to 85°C
M – Military 5V ±10% -55° to 125°C

Capacitance

T_A = 25°C f = 1 MHz
unmeasured pins returned to ground

Symbol	Parameter	Max	Unit
C_ϕ	Clock Capacitance	35	pf
C_{IN}	Input Capacitance	5	pf
C_{OUT}	Output Capacitance	10	pf

Z80A-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C

A.C. Characteristics Z80A-CPU

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition	
Φ	t_c	Clock Period	25	[12]	μsec		[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	[1]	nsec		
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec		
	t_r, t_f	Clock Rise and Fall Time		30	nsec		
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec	C_L = 50pF	
	$t_{F(AD)}$	Delay to Float		90	nsec		
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec		[1] $t_{acm} = t_{w(\Phi H)} + t_f - 65$
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec		[2] $t_{aci} = t_c - 70$
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec		
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec		
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec	C_L = 50pF	[3] $t_{ca} = t_{w(\Phi L)} + t_f - 50$
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec		
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec		[4] $t_{caf} = t_{w(\Phi L)} + t_f - 45$
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec		
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec		[5] $t_{dcm} = t_c - 170$
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec		
	t_{ccf}	Data Stable From $\overline{WR}$	[7]				[6] $t_{dci} = t_{w(\Phi L)} + t_f - 170$
	t_H	Any Hold Time for Setup Time		0	nsec		[7] $t_{cdf} = t_{w(\Phi L)} + t_f - 70$
$\overline{MREQ}$	$t_{DL\Phi(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		85	nsec	C_L = 50pF	
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec		
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec		
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec		[8] $t_{w(\overline{MRL})} = t_c - 30$
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec		[9] $t_{w(\overline{MRH})} = t_{w(\Phi H)} + t_c - 20$
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec	C_L = 50pF	
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec		
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec		
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec		
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		85	nsec	C_L = 50pF	
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		95	nsec		
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		85	nsec		
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		85	nsec		
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		65	nsec	C_L = 50pF	
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		80	nsec		
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		80	nsec		
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec		[10] $t_{w(\overline{WRL})} = t_c - 30$
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		100	nsec	C_L = 50pF	
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		100	nsec		
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		130	nsec	C_L = 50pF	
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		120	nsec		
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec		
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF	
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec		
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec		
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	50		nsec		
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		100	nsec	C_L = 50pF	
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		100	nsec		
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	60		nsec		
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec		
	t_{ml}	M1 Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec		[11] $t_{ml} = 2t_c + t_{w(\Phi H)} + t_f - 65$

NOTES:

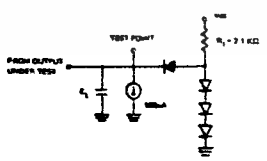
A. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

D. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V ±5%
Add 10nsec delay for each 50pf increase in load up to maximum of 200pf for data bus and 100pf for address & control lines.

E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum



Load circuit for Output

Zilog

COMPONENTS

Z-80® PIO
Z-80A PIO

Product Specification

The Zilog Z-80 product line is a complete set of microcomputer components, development systems and support software. The Z-80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

The Z-80 Parallel I/O (PIO) Interface Controller is a programmable, two port device which provides TTL compatible interfacing between peripheral devices and the Z80-CPU. The Z80-CPU configures the Z80-PIO to interface with standard peripheral devices such as tape punches, printers, keyboards, etc.

Structure

- N-Channel Silicon Gate Depletion Load technology
- 40 Pin DIP
- Single 5 volt supply
- Single phase 5 volt clock
- Two independent 8-bit bidirectional peripheral interface ports with "handshake" data transfer control

eatures

- Interrupt driven "handshake" for fast response
- Any one of the following modes of operation may be selected for either port:
  - Byte output
  - Byte input
  - Byte bidirectional bus (available on Port A only)
  - Bit Mode
- Programmable interrupts on peripheral status conditions.
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.
- Eight outputs are capable of driving Darlington transistors.
- All inputs and outputs fully TTL compatible.

PIO Architecture

A block diagram of the Z80-PIO is shown in figure 1. The internal structure of the Z80-PIO consists of a Z80-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. A typical application might use Port A as the data transfer channel and Port B for the status and control monitoring.

The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in figure 2. The registers include: an 8-bit input register, an 8-bit output register, a 2-bit mode control register, an 8-bit mask register, an 8-bit input/output select register, and a 2-bit mask control register. The last three registers are used only when the port has been programmed to operate in the bit mode.

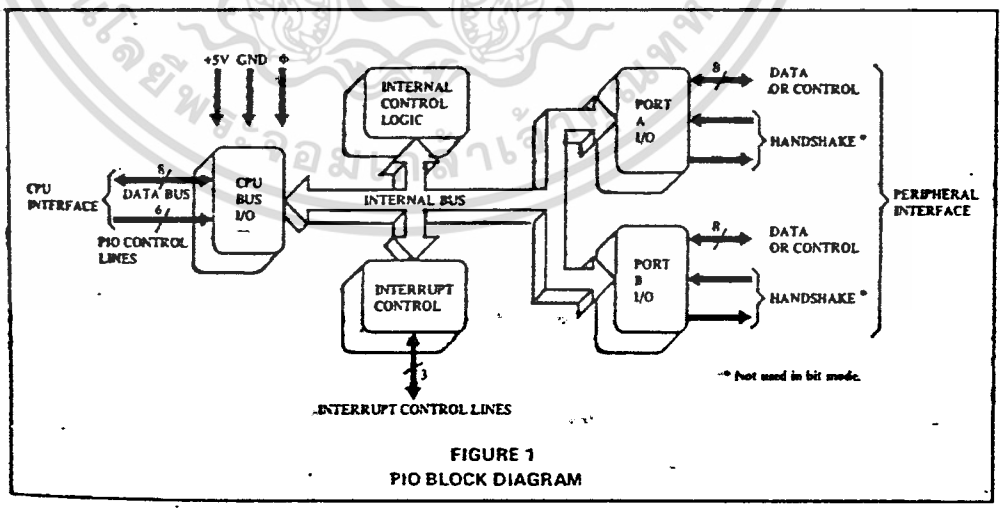


FIGURE 1
PIO BLOCK DIAGRAM

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MOC3020
MOC3021
MOC3022
MOC3023

6-Pin DIP Optoisolators
Triac Driver Output

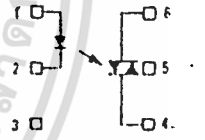
These devices consist of gallium-arsenide infrared emitting diodes, optically coupled to a silicon bilateral switch.
They are designed for applications requiring isolated triac triggering.

- UL Recognized File Number E54915
- Output Driver Designed for 240 Vac Line
- V_{ISO} Isolation Voltage of 7500 V Peak
- Similar to MOC3010 and MOC3011
- Standard 6-PIN Plastic DIP
- VDE approved per standard 0883/6.80 (Certificate number 41853), with additional approval to DIN IEC380/VDE0806, IEC435/VDE0805, IEC65/VDE0860, VDE110b, covering all other standards with equal or less stringent requirements, including IEC204/VDE0113, VDE0160, VDE0832, VDE0833, etc.
- Special lead form available (add suffix "1" to part number) which satisfies VDE0883 6.80 requirement for 8 mm minimum creepage distance between input and output solder pads.
- Various lead form options available. Consult "Optoisolator Lead Form Options" data sheet for details.

6-PIN DIP
OPTOISOLATORS
TRIAC DRIVER OUTPUT

CASE 730A-02
PLASTIC

COUPLER SCHEMATIC

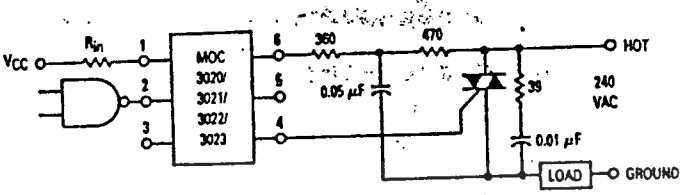


1 ANODE
2 CATHODE
3 NC
4 MAIN TERMINAL
5 SUBSTRATE, DO NOT CONNECT
6 MAIN TERMINAL

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INFRARED EMITTING DIODE			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
Total Power Dissipation @ T_A = 25°C Negligible Power in Triac Driver Derate above 25°C	P_D	100 1.33	mW mW/°C
OUTPUT DRIVER			
Off-State Output Terminal Voltage	V_{DRM}	400	Volts
Peak Repetitive Surge Current (PW = 1 ms, 120 pps)	I_{TSM}	1	A
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	300 4	mW mW/°C
TOTAL DEVICE			
Isolation Surge Voltage (1) (Peak ac Voltage, 60 Hz, 5 Second Duration)	V_{ISO}	7500	Vac
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	330 4.4	mW mW/°C
Junction Temperature Range	T_J	−40 to +100	°C
Ambient Operating Temperature Range	T_A	−40 to +85	°C
Storage Temperature Range	T_{stg}	−40 to +150	°C
Soldering Temperature (10 s)	—	260	°C

(1) Isolation surge voltage, V_{ISO}, is an internal device dielectric breakdown rating.



*This optoisolator should not be used to drive a load directly. It is intended to be a trigger device only.

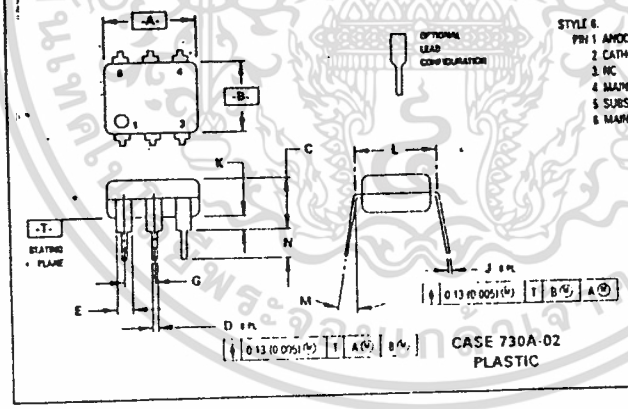
Additional information on the use of optically coupled triac drivers is available in Application Note AN-780A.

In this circuit the "hot" side of the line is switched and the load connected to the cold or ground side.

The 39 ohm resistor and 0.01 μF capacitor are for snubbing of the triac, and the 470 ohm resistor and 0.05 μF capacitor are for snubbing the coupler. These components may or may not be necessary depending upon the particular triac and load used.

Figure 8. Typical Application Circuit

OUTLINE DIMENSIONS



STYLE 6:
PIN 1 ANODE
2 CATHODE
3 NC
4 MAIN TERMINAL
5 SUBSTRATE
6 MAIN TERMINAL

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIM L TO CENTER OF LEAD WHEN FORMED PARALLEL.

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	6.13	6.99	0.370	0.350
B	6.10	6.60	0.240	0.260
C	2.93	5.08	0.115	0.200
D	0.41	0.50	0.016	0.020
E	1.02	1.77	0.040	0.070
G	2.54 BSC		0.100 BSC	
J	0.21	0.30	0.008	0.012
K	0.38	2.54	0.015	0.100
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	2.54	3.81	0.100	0.150

MOC3020, MOC3021, MOC3022, MOC3023

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
INPUT LED						
Reverse Leakage Current (V_R = 3 V)		I_R	—	0.05	100	μA
Forward Voltage (I_F = 10 mA)		V_F	—	1.15	1.5	Volts
OUTPUT DETECTOR (I_F = 0 unless otherwise noted)						
Peak Blocking Current, Either Direction (Rated V_{DRM}, Note 1)		I_{DRM}	—	10	100	nA
Peak On-State Voltage, Either Direction (I_{TM} = 100 mA Peak)		V_{TM}	—	1.8	3	Volts
Critical Rate of Rise of Off-State Voltage (Figure 7, Note 2)		dv/dt	—	10	—	V/μs
COUPLED						
LED Trigger Current, Current Required to Latch Output (Main Terminal Voltage = 3 V, Note 3)	MOC3020	I_{FT}	—	15	30	mA
	MOC3021		—	8	15	
	MOC3022		—	—	10	
	MOC3023		—	—	5	
Holding Current, Either Direction		I_H	—	100	—	μA

Notes: 1. Test voltage must be applied within dv/dt rating.
2. This is static dv/dt. See Figure 7 for test circuit. Commutating dv/dt is a function of the load-driving thyristor(s) only.
3. All devices are guaranteed to trigger at an I_F value less than or equal to max I_{FT}. Therefore, recommended operating I_F lies between max I_{FT} (30 mA for MOC3020, 15 mA for MOC3021, 10 mA for MOC3022, 5 mA for MOC3023) and absolute max I_F (60 mA).

TYPICAL ELECTRICAL CHARACTERISTICS

T_A = 25°C

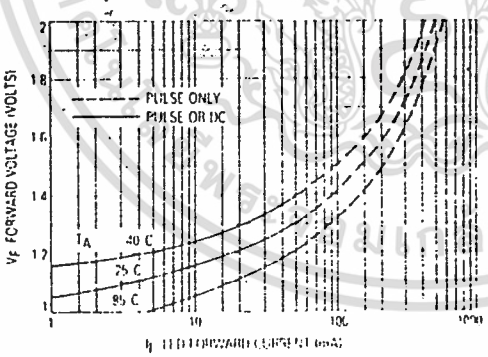


Figure 1. LED Forward Voltage versus Forward Current

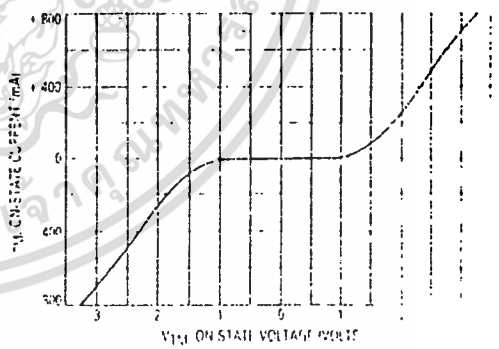


Figure 2. On-State Characteristics

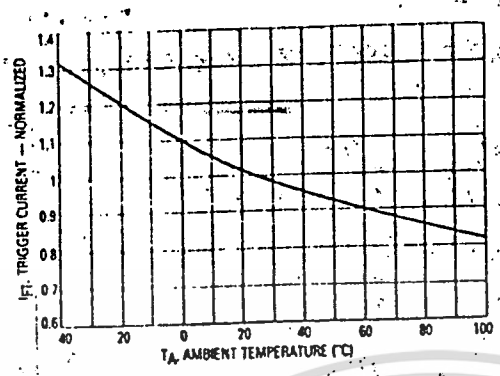


Figure 3. Trigger Current versus Temperature

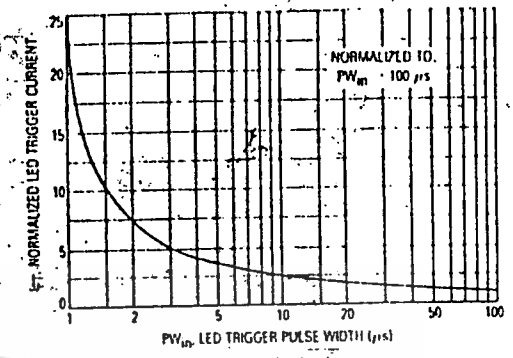


Figure 4. LED Current Required to Trigger versus LED Pulse Width

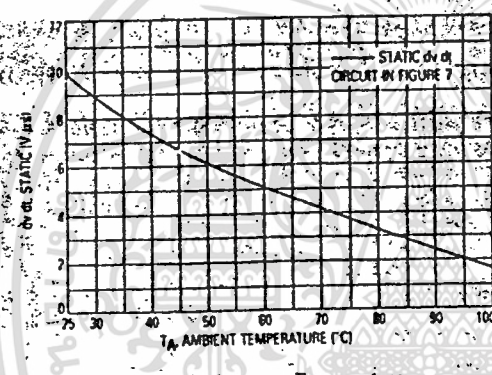


Figure 5. dv/dt versus Temperature

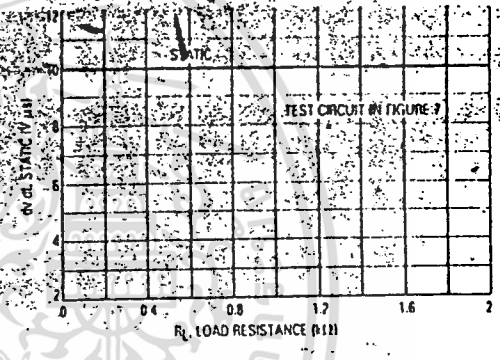
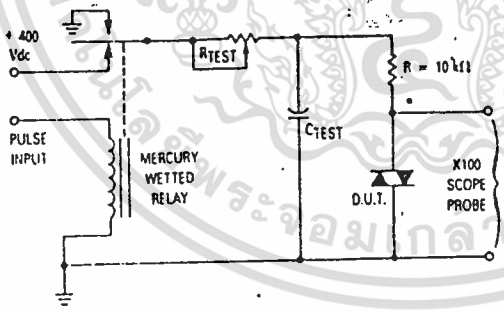


Figure 6. dv/dt versus Load Resistance



1. The mercury wetted relay provides a high speed repeated pulse to the D.U.T.
2. 100× scope probes are used, to allow high speeds and voltages.
3. The worst case condition for static dv/dt is established by triggering the D.U.T. with a normal LED input current, then removing the current. The variable RTEST allows the dv/dt to be gradually increased until the D.U.T. continues to trigger in response to the applied voltage pulse, even after the LED current has been removed. The dv/dt is then decreased until the D.U.T. stops triggering. τRC is measured at this point and recorded.

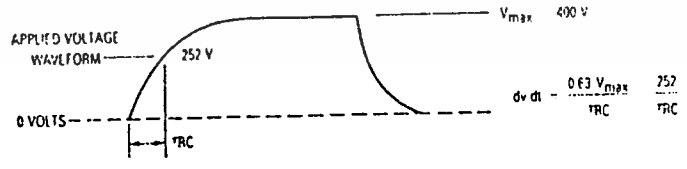
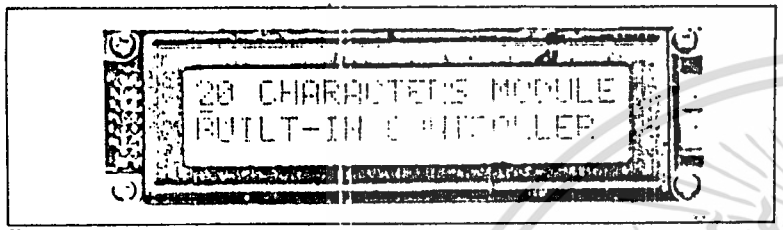


Figure 7. Static dv/dt Test Circuit

HDM-20216H



20 CHARACTERS X 2 LINES MODULE

PHYSICAL DATA

Module size	120.0W x 36.0H x 10.0D mm
Min. view area	88.2W x 20.0H mm
Character construction	5 x 7 dots
Character size	3.4W x 5.15H mm
Character pitch	4.2 mm
Dot size	0.6W x 0.65H mm
Weight	about 28g

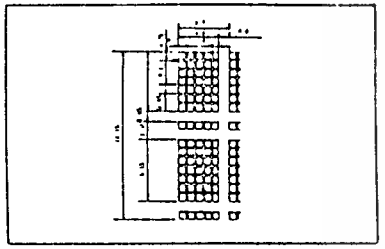
ABSOLUTE MAXIMUM RATINGS

	min	max
Power supply for logic (V_{DD} - V_{SS})	0	7.0 V
Power supply for LCD (V_{DD} - V_L)	0	13.5 V
Input voltage (V_{IN})	V_{SS}	V_{DD} V
Operation temperature (T_{OP})	0°C	50°C
Storage temperature (T_{STG})	-20°C	70°C

ELECTRICAL CHARACTERISTICS (V_{DD} =5.0±0.25V 25°C)

Input high voltage (V_{IH})	2.2V min.
Input low voltage (V_{IL})	0.6V max.
Output high voltage (V_{OH}) . ($-I_{OH}$=0.205mA)	2.4V min.
Output low voltage (V_{OL}) .. (I_{OL}=1.2mA)	0.4V max.
Power supply current (I_{DD}) . (V_{DD}=5.0V)	1.2mA typ. 2.3mA max.
Drive method	1/16 Duty
Power supply for LCD drive (V_{DD} - V_L)	
at Ta = 0°C	4.6V typ.
Ta = 25°C	4.4V typ.
Ta = 50°C	3.6V typ.

DISPLAY PATTERN

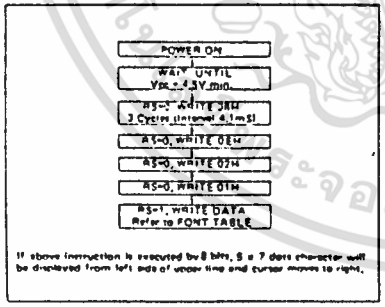


PIN CONNECTIONS

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0 V	Power supply
2	V_{DD}	–	5 V	
3	V_L	–	–	
4	RS	H/L	H: Data input L: Instruction data input	
5	R/W	H/L	H: Data read L: Data write	
6	E	H→L	Enable signal	
7	DO	H/L		
8	D1	H/L		
9	D2	H/L		
10	D3	H/L		
11	D4	H/L		Data bus line*
12	D5	H/L		
13	D6	H/L		
14	D7	H/L		

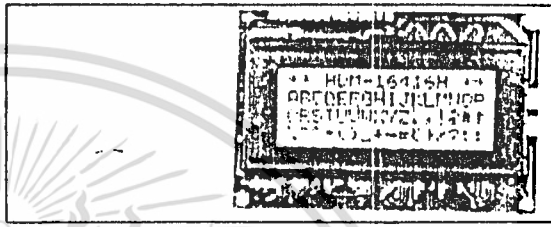
* In case of 4 bits instruction, data is transferred by twice using only 4 buses of D4-D7, and D0-D3 are not used. First operation is higher order 4 bits and second is lower 4 bits of 8 bits, but in case of 8 bits instruction, data is transferred by data bus of D0-D7.

TEST PROCEDURE



If above instruction is executed by 8 bits, 5 x 7 dots character will be displayed from left side of upper line and cursor moves to right.

HDM-16416H



16 CHARACTERS X 4 LINES MODULE

PHYSICAL DATA

Module size	87.0W x 60.0H x 10.0D mm
Min. view area	61.8W x 25.2H mm
Character construction	5 x 7 dots
Character size	2.95W x 4.15H mm
Character pitch	3.55 mm
Dot size	0.55 x 0.55 H mm
Weight	about 40g

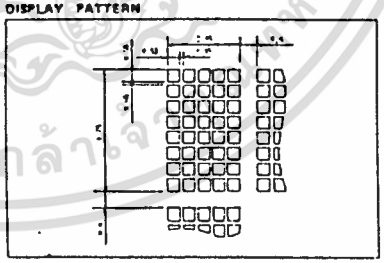
ABSOLUTE MAXIMUM RATINGS

	min	max
Power supply for logic (V_{DD} - V_{SS})	0	7.0 V
Power supply for LCD (V_{DD} - V_L)	0	13.5 V
Input voltage (V_{IN})	V_{SS}	V_{DD} V
Operation temperature (T_{OP})	0°C	50°C
Storage temperature (T_{STG})	-20°C	70°C

ELECTRICAL CHARACTERISTICS (V_{DD} =5.0±0.25V 25°C)

Input high voltage (V_{IH})	2.2V min.
Input low voltage (V_{IL})	0.6V max.
Output high voltage (V_{OH}) . ($-I_{OH}$=0.205mA)	2.4V min.
Output low voltage (V_{OL}) .. (I_{OL}=1.2mA)	0.4V max.
Power supply current (I_{DD}) . (V_{DD}=5.0V)	2.0mA typ. 3.0mA max.
Drive method	1/16 Duty
Power supply for LCD drive (V_{DD} - V_L)	
at Ta = 0°C	4.6V typ.
Ta = 25°C	4.4V typ.
Ta = 50°C	3.6V typ.

DISPLAY PATTERN

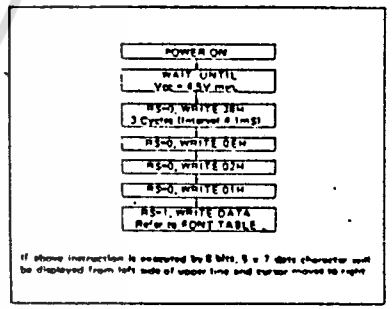


PIN CONNECTIONS

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0 V	Power supply
2	V_{DD}	–	5 V	
3	V_L	–	–	
4	RS	H/L	H: Data input L: Instruction data input	
5	R/W	H/L	H: Data read L: Data write	
6	E	H→L	Enable signal	
7	DO	H/L		
8	D1	H/L		
9	D2	H/L		
10	D3	H/L		
11	D4	H/L		Data bus line*
12	D5	H/L		
13	D6	H/L		
14	D7	H/L		

* In case of 4 bits instruction, data is transferred by twice using only 4 buses of D4-D7, and D0-D3 are not used. First operation is higher order 4 bits and second is lower 4 bits of 8 bits, but in case of 8 bits instruction, data is transferred by data bus of D0-D7.

TEST PROCEDURE



If above instruction is executed by 8 bits, 5 x 7 dots character will be displayed from left side of upper line and cursor moves to right.

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่องนี้ได้แนวความคิดจาก อาจารย์ อรรถสิทธ์ หล่าสกุล ทั้งยังได้แนะนำแนวทางการทำงานของเครื่องและสนับสนุนทางด้านเครื่องมือและอุปกรณ์มาโดยตลอด จึงขอกราบขอบพระคุณ อาจารย์ อรรถสิทธ์ หล่าสกุล เป็นอย่างสูงไว้ ณ ที่นี้ด้วย

ขอขอบคุณ อาจารย์ภาควิชาเทคนิคอุตสาหกรรมที่ให้ความร่วมมือและช่วยเหลืออำนวยความสะดวกในการทำปริญญานิพนธ์ด้วยดีมาตลอด

ท้ายนี้ขอกราบพระคุณแม่บิดามารดา ที่ให้การสนับสนุนในทุกๆ ด้านแก่ผู้ทำปริญญานิพนธ์มาโดยตลอดไว้ ณ ที่นี้ด้วย

หนังสืออ้างอิง

1. Curtis D.Johnson "Microprocessor-Based Process Control" Pretice-Hall , 1984
2. Gio Wiederhold "Database Design" Mc Graw-Hill , 1977
3. John Hindmarsh "Electrical Machines and Their Applications" 4th Edition , 1977
4. Hard book MCS-51 , Intel Corportion , 1985
5. เซมิคอนดัคเตอร์อิเลคทรอนิคส์ เล่ม 98 "RTC ระบบรีไทม์คล็อก" กุมภาพันธ์ 2533
6. ยืน ภู่วรวรรณ,วัฒนา เชียงกูล " Z-80 MICROPROCESSOR " ซีเอ็ด,2521
7. "คู่มือ การพันหม้อแปลง/มอเตอร์" ซีเอ็ด,2528